UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Cooking Code: Juego para Aprender a Programar con Realidad Virtual

Autor: Iván Gómez Jiménez

Director: Dra. Inmaculada Rodríguez Santiago
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, September 14, 2020

Abstract (English)

The present project focuses on creating an educational game that allows kids, between the ages of 12 and 16, to get involved in the programming world.Therefore, this game belongs to the serious games's scoupe, wich have as main goal the education. The game is based on the metaphor of a Burger, set in a futuristic era where robots and humans coexist as equals. The player will be the chef of the restaurant, having to learn to read the orders that arrive and carry them out successfully. The main difficulty is that the requests do not arrive in natural language, but in a kind of pseudo-code. This context allows the user to find himself performing simple actions known as making a hamburger, focusing his attention on trying to understand the steps to be carried out through basic, conditional and iterative programming sentences. The chosen platform has been virtual reality, specifically the Oculus Quest. Apart from being a novel technology, it provides very important benefits such as immersion or achieving much more realistic interactions with the environment. As a result of all this, it seeks to create an innovative, fun game that puts the world of programming within the reach of new generations.

Abstract (Castellano)

El presente proyecto se centra en la creación de un juego educativo que permita a niños, de entre 12 y 16 años, introducirse a la programación. Por lo tanto, este juego se sitúa en el ámbito de los serious games, los cuales tienen como objetivo principal la formación. El juego se basa en la metáfora de una hamburguesería, ambientada en una época futurista dónde robots y humanos conviven como iguales. El jugador será el cocinero del restaurante, teniendo que aprender a leer los pedidos que lleguen y realizándolos con éxito. La principal dificultad está en que los pedidos no llegan en lenguaje natural, sino en una especie de pseudocódigo. Este contexto permite al usuario encontrarse realizando acciones sencillas y conocidas como hacer una hamburguesa, poniendo el foco de su atención en tratar de entender los pasos a realizar a través de sentencias de programación básicas, condicionales e iterativas. La plataforma escogida ha sido la realidad virtual, concretamente las Oculus Quest. A parte de ser una tecnología novedosa, proporciona beneficios muy importantes como la inmersión o conseguir interacciones con el entorno mucho más realistas. Como resultado de todo ello, se busca crear un juego innovador, divertido y que ponga el mundo de la programación al alcance de las nuevas generaciones.

Abstract(Català)

L'actual projecte es centra en la creació d'un joc educatiu que permeti a nens, d'entre 12 i 16 anys, introduir-se a la programació. Per això, aquest joc es situa a l'àmbit dels serious games, els quals tenen com a objectiu principal la formació. El joc es basa en la metàfora d'una hamburgueseria, ambientada a una època futurista on robots i humans conviuen com iguals. El jugador serà el cuiner del restaurant, aprenent a llegir les comandes que arriben i realitzant-les amb èxit. La principal dificultat està que les comandes no arriben en llenguatge natural, sinó en una mena de pseudocodi. Aquest context permet a l'usuari fer accions simples i conegudes com fer una hamburguesa, posant l'atenció en tractar d'entendre els passos a seguir a través de sentències de programació bàsiques, condicionals i iteratives. La plataforma seleccionada ha sigut la realitat virtual, concretament les Oculus Quest. A banda de ser una tecnologia molt nova, proporciona beneficis molt importants com la immersió o aconseguir interaccions amb l'entorn

molt més realistes. Com a resultat de tot això, es vol crear un joc innovador, divertir i que posi el món de la programació a l'abast de les noves generacions.

Índice

Índice de figuras

1 Introducción

Este trabajo tiene una clara vocación educacional. En mi opinión, la enseñanza se debe enfocar con nuevas metodologías. Concretamente, se pretende analizar y explorar las ventajas de aprender mediante los serious games, en este caso, utilizando como plataforma la realidad virtual.

Los Serious Games son juegos o experiencias que tienen como objetivo principal la formación, sin olvidar la diversión como componente fundamental. Su propósito se centra en lo que se denomina como training educacional. Forman una muestra más, de como los avances tecnológicos pueden y deben ser integrados, con responsabilidad, en nuestra sociedad. A día de hoy, cualquier niño es nativo digital, tiene absolutamente naturalizados los mecanismos y el uso de teléfonos, tabletas, ordenadores ... Es por eso que estas metodologías, como lo son los Serious Games, pueden ser tan beneficiosas, ocupando un lugar muy importante en los sistemas educativos.

Por otro lado, la realidad virtual es una novedosa plataforma con un potencial muy grande en el ámbito educacional. No es sólo el atractivo de ser una tecnología emergente sino que además, forma una combinación muy buena con los serious games. El poder tener la oportunidad de explorar y explotar los efectos de la inmersión, u otros beneficios de la realidad virtual aplicados a dichas experiencias educativas, ha sido una gran motivación para realizar el trabajo. Más concretamente, las gafas RV (Realidad virtual) han sido las Oculus Quest, las cuales se introducirán en posteriores puntos del documento (Ver sección 4).

En este trabajo se propone el diseño e implementación de un juego que permita, a estudiantes de entre 12 y 16 años, aprender la lógica de la programación. El juego se ambienta en una época futura, dónde máquinas y humanos se han equiparado, produciendo incluso una mezcla en el lenguaje. El usuario tendrá el papel del nuevo cocinero de una hamburguesería, teniendo que realizar los pedidos que le van llegando. No obstante, el lenguaje ha sufrido muchos cambios, y los pedidos de las hamburguesas llegarán en forma de pseudocódigo. Por lo tanto, tendrá que entender el código y montar la hamburguesa correctamente antes que acabe el día.

Durante este trabajo, he podido aplicar conocimientos adquiridos en diferentes asignaturas. A lo largo de toda la etapa de diseño, me he basado en muchos conceptos vistos en la asignatura de *Factors Humans*. Además, he aprovechado la experiencia que gané realizando un videojuego en *Projecte Integrat de Software*. Para la base de datos y su acceso, he utilizado conocimientos adquiridos en *Enginyeria del Software* y por último, los patrones de diseño aprendidos en *Disseny de Software*.

2 Objetivos

2.1 Objetivo general

El objetivo principal del proyecto es crear un juego, llamado *Cooking Code*, que permita a los niños de entre 12 y 16 años entrar al mundo de la programación. Aprender a programar, aunque se expliquen las instrucciones básicas y su comportamiento, no es fácil. Existe un componente práctico y de resolución de problemas que es el que acaba de formar al estudiante. Jugando a *Cooking Code*, se pretende que el jugador desarrolle:

- Comprensión de los componentes de un programa: variables, tipos de datos, identaciones, oper-

adores, expresiones y palabras clave.

- Conocimiento y habilidades en el uso de las instrucciones básicas: secuenciales, condicionales e iterativas.

Por lo tanto, el juego se localiza en ese componente práctico del aprendizaje de la programación.

La realización de dichos objetivos implica el desarrollo del pensamiento computacional por parte del usuario. Esta forma de razonar incluye las siguientes características:

- Organizar y analizar lógicamente la información.
- Representar la información a través de abstracciones como modelos y simulaciones.
- Desarrollo del pensamiento algorítmico, basado en establecer una serie de pasos ordenados que permiten llegar a una solución efectiva y eficiente, tanto en pasos realizados como en recursos utilizados.

2.2 Objetivos específicos

El objetivo principal se desglosa en los siguientes objetivos específicos:

- Realizar un análisis de la competencia: Permitirá estudiar el mercado de productos similares, lo que tienen que ofrecer y lo que no, aquello que ha funcionado, o que puede mejorarse.
- Análisis de herramientas de desarrollo: Se ha de realizar un trabajo de familiarización tanto con el entorno de desarrollo (Unity, C#)[10] como con la plataforma (RV, Oculus Quest).
- Análisis de los usuarios: Con un adecuado análisis de los usuarios, podremos diseñar el juego acorde a sus características, necesidades y motivaciones.
- Diseño e implementación del juego: Es una parte fundamental del proceso, creando un juego que permita aprender a la vez que divierte.
- Diseño e implementación del backend: Para que los usuarios guarden sus puntuaciones y progresos, se ha de diseñar una base de datos y permitir acceder a ella a través de una API (Interfaz de programación de aplicaciones).

3 Análisis de la competencia

En este apartado se realizará el análisis de la competencia. Dicho estudio, se basa en la descripción de mecánicas y funcionalidades presentes en aplicaciones similares al proyecto actual. Éstas se pueden dividir en dos grupos: según se basen o no en realidad virtual.

El objetivo es aprender tanto de los aspectos positivos como de los negativos de dichos antecedentes. Describiendo además qué puede aportar *Cooking Code* respecto al resto de juegos del mismo sector.

3.1 No basados en realidad virtual

3.1.1 Scratch

Probablemente la aplicación más conocida sea Scratch. Scratch es un lenguaje de programación visual creado por el MIT (ver Figura 1), centrado en facilitar el aprendizaje siendo lo más intuitivo posible. En lugar de código, se nos presentan diferentes piezas, similares a un puzzle, que podemos ir encajando. A la vez que construimos dicho "rompecabezas", se irán desarrollando diferentes secuencias de instrucciones del programa que formarán el proyecto. Al acceder a la aplicación, podemos ver algo similar a un entorno de desarrollo. En un menú lateral, aparecen todos los bloques de acciones seleccionables, pudiendo arrastrarlos hacía la pantalla que constituye "nuestro código". Mediante las restricciones en las formas, se entienden rápidamente las relaciones de inclusión o dependencia entre las diferentes instrucciones. Además, cada tipo de bloque es de un color bien diferenciado, lo que ayuda a localizarlos de una forma rápida y clara.

Figura 1: Pantalla en un proyecto de Scratch (Fuente: `https://ladiversiva.com/`)

Además, se nos proporciona un feedback rápido y atractivo. Con un simple clic, y sin cambiar de pestaña, podemos ver el resultado de la ejecución. Este aspecto, junto a todo el material proporcionado, permite fomentar continuamente la creatividad del usuario. Comenzar a utilizarlo puede resultar sencillo, incluso dando una cierta sensación de simplicidad. Gracias a los diferentes materiales didácticos del propio portal, y a la curva de dificultad tan suave y gratificante que presenta al inicio, es muy sencillo obtener buenos resultados.

En cuanto a la influencia sobre el proyecto actual, Scratch ha permitido ver que es posible enseñar programación de una forma divertida. Las diferencias más importantes entre el presente juego y Scratch, se encuentran en cómo el usuario interacciona con el código. Por una parte, Scratch permite al jugador combinar diferentes instrucciones, dándole más importancia a la creatividad. Por otro lado, en *Cooking Code* el código viene ya dado, y es el usuario el que lo ha de interpretar y ejecutar. Además, el proyecto actual tiene un contexto, una historia, que lo engloba y le da sentido. En cambio, Scratch es más una herramienta para crear, no tanto una experiencia por si mismo. Aún considerando éstas diferencias, será una fuente de inspiración durante este trabajo.

3.1.2 Human Resource Machine

Juego de puzzles desarrollado por Tomorrow Corporation, basado principalmente en la programación visual. Consta de aproximadamente 40 niveles a resolver, cada uno considerado un año de trabajo del protagonista que nos acompañará en la historia.

En la trama, seremos un nuevo empleado de una gran empresa, que ha de ir subiendo en la escala de poder, representada por pisos de un ascensor.

Cada nivel, será una nueva oportunidad en un puesto de trabajo, dándonos acceso a otros puesto más elevados en la organización. Además, diferentes historias se entrelazan formando una trama rica y trabajada, sobretodo considerando la temática del título. En los puzzles, el usuario tiene que crear una secuencia de instrucciones que guíen al protagonista en su desempeño. Cada vez, la tarea a realizar será diferente, y con una complejidad mayor conforme avancen los niveles.

Al principio, ya podemos ver como se introduce el concepto de secuencia de instrucciones. Además, el bucle es representado de una forma original, haciendo al usuario arrastrar una flecha hasta donde quiere que se realice (ver Figura 2). Así, en todo momento se mantiene la referencia visual del flujo de ejecución que, en determinadas ocasiones, puede resultar complejo o confuso.

Figura 2: Nivel en Human Resource Machine (Fuente: `https://bit.ly/2ZzVhKZ`)

Lo que hace a este juego interesante es ver la programación visual aplicada a un contexto distinto a Scratch, y de una forma tan diferente. Se ha conseguido incluir de forma exitosa el componente educativo sin privarlo de un contexto y una historia. El usuario, a través de la repetición, aprende e interioriza el pensamiento computacional, y la historia y los objetivos del juego hacen que no resulte monótono.

En definitiva, es una referencia muy clara de lo que se quiere conseguir en este proyecto, aprender a programar de forma divertida.

3.2 Basados en realidad virtual

3.2.1 CoSpaces

CoSpaces es una plataforma para crear, explorar y compartir mundos tridimensionales. Su principal función es, como veíamos con Scratch, permitir a usuarios inexpertos en programación desarrollar y expresar sus ideas de una forma simple y clara. Actualmente, estos mundos se pueden crear y visualizar desde cualquier dispositivo: navegador, tableta, teléfono móvil y con gafas de realidad virtual.

Los recursos educativos en realidad virtual suelen ser escasos, basándose la mayoría en la visualización: un recorrido por una ciudad, una visita a un museo, visualizaciones de estructuras tridimensionales, etc. Esto es debido a la dificultad que implica programar contenidos interactivos en dicha plataforma. La mayoría de las herramientas son profesionales, utilizadas en entornos de desarrollo de videojuegos, véase Unity o Unreal, cuya curva de aprendizaje es demasiado larga y costosa. Ese vacío es el que viene a ocupar CoSpaces, permitiendo crear estos materiales y dotando a dichos recursos educativos de una calidad e interacción muy superior a la convencional.

Existen dos entornos de programación para realizar estos proyectos: Blockly (programación visual) y JavaScript. La opción más interesante es Blockly (ver Figura 3). Si el usuario conoce Scratch, o herramientas similares, no le supondrá esfuerzo alguno familiarizarse con dicho entorno. De forma muy similar a la programación visual vista antes con Scratch, tendremos que ir encajando bloques para crear la lógica de un programa, es decir, una secuencia de instrucciones. Además, se añade un grado de complejidad, debido a nuevas variables o acciones relacionadas con los mundos en tres dimensiones. Hace un gran esfuerzo por integrarse en el mundo educativo, desarrollando herramientas interesantes en dicho ámbito. En CoSpaces, el docente puede ver en tiempo real las creaciones de los alumnos, así como compartir proyectos propios en otros dispositivos.

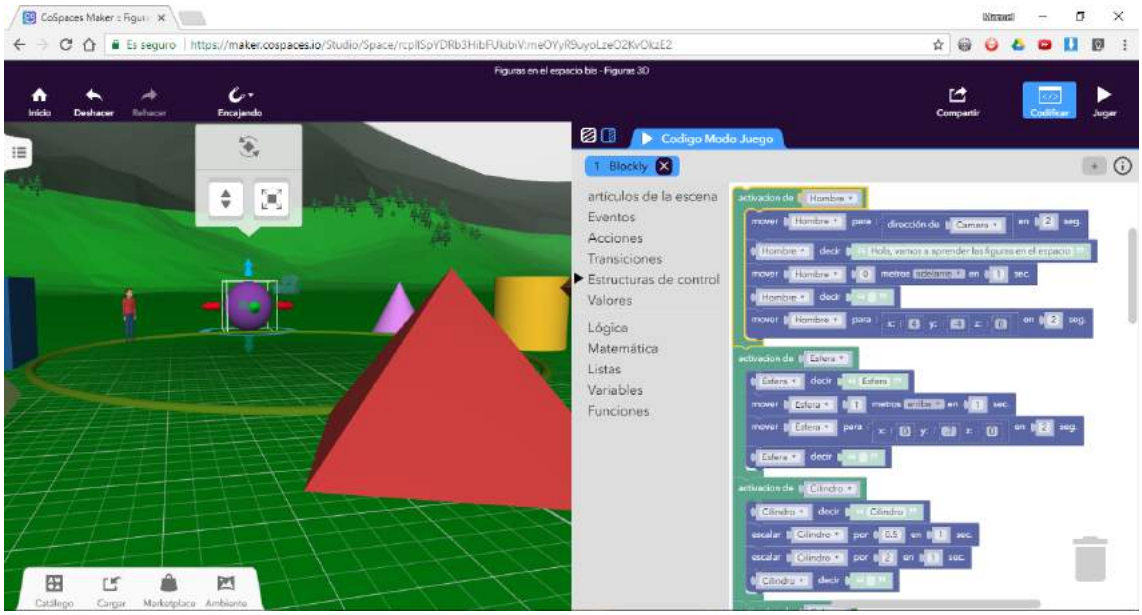

Figura 3: Programación visual con Blockly (Fuente: `http://manuelalmonacid.blogspot.com/`)

Al igual que se había concluido con Scratch, ésta es una herramienta potente y compleja, que permite ser utilizada por usuarios inexpertos en programación. Cospaces ha inspirado algunas ideas de este proyecto, como por ejemplo, la programación visual en la realidad virtual. Durante el proyecto, pudimos hablar con uno de los embajadores de dicha plataforma, Jesus Arbues, quien nos explicó sus experiencias en el mundo docente con la tecnología y CoSpaces. Jesús es un docente de una escuela de secundaria en Barcelona, concretamente trabaja con el departamento de artes plásticas. Recientemente, incluyó en sus clases recursos basados en realidad virtual, que permitían hacer la clase más amena y efectiva. Pese a ello, se muestra reticente al uso de la realidad virtual en el entorno educativo por los costes de los dispositivos como las Oculus Quest. En la entrevista, se mostró realmente entusiasmado con poder realizar sus propios proyectos, y permitir a sus alumnos tener clases de una mayor calidad y diversión (ver apéndice B).

3.2.2 Job Simulator

Job Simulator (ver Figura 4) es un juego de realidad virtual desarrollado y publicado por Owlchemy Labs. La experiencia virtual permite a los usuarios participar en simulaciones de trabajos reales, con un toque cómico y divertido. Entre las ocupaciones podemos encontrar: mecánico, chef, oficinista o cajero. En dicho universo, nos encontraremos siempre acompañados de ordenadores caracterizados, los cuales nos proporcionarán los trabajos e instrucciones a realizar. Generalmente, estas tareas suelen ser realistas, pero se combinan con otras más cómicas, dándole mucho más dinamismo. Por ejemplo, en el trabajo de oficinista, el jugador tendrá que evaluar a nuevos empleados, atender llamadas... pero también podrá comer donuts, compartir fotos o participar en otras tareas. Visualmente, este juego es muy parecido, por temática y por estilo, a lo que se desea hacer en este proyecto. De hecho, será uno de los principales referentes en el desarrollo y el modelado del entorno de *Cooking Code*. Con un estilo desenfadado y relajado, consigue hacer que tareas realmente estresantes se vean como agradables y divertidas. Además, la interacción con el entorno pretende ser realista, lo cuál permite entender y aprender los controles de RV de una forma rápida y sencilla.

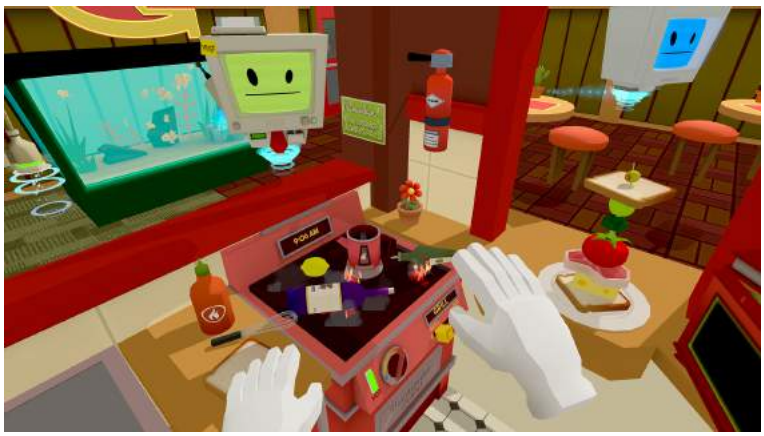

Figura 4: Cocina de Job Simulator (Fuente: `https://bit.ly/32oDF6z`)

Aunque Job Simulator no comparta el objetivo principal de este proyecto, el propósito educativo, permite situar una experiencia de realidad virtual en un entorno, la cocina, muy similar al deseado en este proyecto.

4 Tecnologías

Oculus Quest

La historia de las gafas de realidad virtual va ligada con la compañía Oculus VR. Fundada en julio de 2012 en Estados Unidos, es una empresa especializada en productos de software y hardware de realidad virtual. En abril de 2012 anunció su primer producto, llamado Oculus Rift, unas gafas de realidad virtual que permitían una experiencia muy novedosa. La compañía lanzó mediante Kickstarter una campaña para financiar dicho proyecto. La estrategia tuvo éxito, ya que recaudó 2.4 millones de dolares, hasta diez veces más del objetivo original. A principios de 2014, atraído por el proyecto, Mark Zuckerberg decidió adquirir la empresa por 2,3 mil millones de dólares.

En el evento Oculus Connect de 2018, fue anunciada la primera generación de productos de la compañía, formada por Oculus Rift, Oculus Go y Oculus Quest. Oculus Quest[7] (ver Figura 5), las últimas en ser presentadas, son unas gafas de realidad virtual que equipan el procesador móvil Snapdragon 835. Si bien es cierto que no pueden ser tan potentes como las anteriores, que iban conectadas a un PC, estás no requieren de un ordenador para su uso, haciéndolas mucho más cómodas e independientes. El diseño de las gafas es más ligero y cómodo de usar, teniendo un peso mucho mejor distribuido. Además, incluye cuatro cámaras de gran angular que permiten el seguimiento posicional, proporcionando seis grados de libertad. Los seis grados de libertad hacen referencia al movimiento en un espacio tridimensional, es decir, el usuario podrá moverse hacia delante/atrás, arriba/abajo, izquierda/derecha sobre tres ejes perpendiculares.

Figura 5: Oculus Quest (Fuente: `https://www.oculus.com/quest/`)

Estás particularidades hacen de las Oculus Quest unas gafas muy adecuadas para realizar un proyecto educativo. En este tipo de proyectos, normalmente las gafas sufren desplazamientos a clases u otros espacios, siendo su montaje y transporte una parte muy importante. Pese a su limitada potencia, lo cuál tiene menos preferencia en nuestro tipo de juego, supone una gran ventaja no tener que cargar con una televisión, un PC, todos los cables o problemas que puedan aparecer.

Los controladores de las Oculus Quest (ver Figura 6) son realmente ergonómicos, ajustándose muy bien a la mano. Contamos con los 4 botones clásicos de cualquier otro controlador convencional, junto a los dos joysticks. Además, la circunferencia superior permite detectar la altura o movimiento del dedo colocado

en esa posición. Siguiendo con el paralelismo de los controladores convencionales, encontramos los dos gatillos en una posición un tanto diferente. Por un lado, uno que queda justo al cierre de la mano. El otro, queda en la cara frontal del mando, pese a seguir siendo fácilmente accesible.

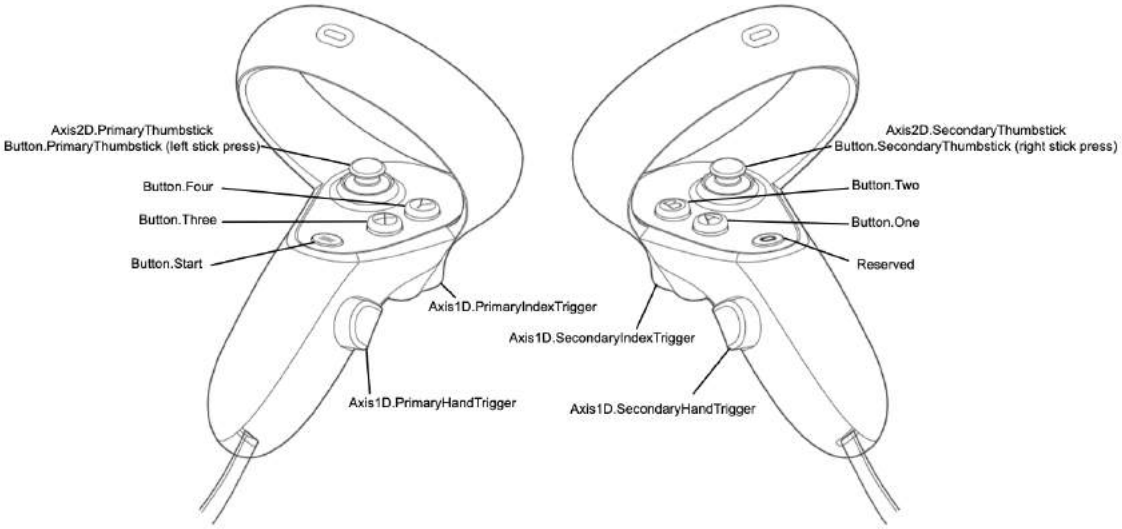

Figura 6: Controladores Oculus Quest (Fuente: `https://ocul.us/3itUnqR`)

Unity

Unity[9] será el motor de videojuegos utilizado en este proyecto. Creado por Unity Technologies, está disponible como plataforma de desarrollo para Windows, Mac OS y Linux. Unity Technologies empezó en 2004, cuando sus creadores decidieron cambiar la dinámica de la compañía tras el fracaso de uno de sus juegos. Aunque dicho juego no había tenido la acogida esperada, el equipo había creado para su desarrollo herramientas muy potentes. De esta forma, pretendían crear con esa base un motor de videojuegos que pequeñas y grandes empresas pudieran utilizar por igual. Con el paso del tiempo, los desarrolladores independientes no tardaron en convertir Unity en uno de los motores de videojuegos más utilizados. Las versiones actuales son compatibles con muchísimas plataformas: PC, Android, PlayStation, Xbox, Wii...

Dada su popularidad, la facilidad de acceder tanto a la herramienta como a sus recursos y la compatibilidad con la plataforma de Oculus Quest, hizo que Unity fuese una elección muy sencilla. Además, pone a disposición de cualquier desarrollador paquetes de soporte que incluyen físicas, interacciones básicas y muchas otras funcionalidades que aceleran todo el proceso de desarrollo.

NodeJS

Node.js[5] es un entorno de programación diseñado para crear aplicaciones web escalables, normalmente del lado del servidor. El lenguaje utilizado es JavaScript, utilizando una arquitectura orientada a eventos, de entrada/salida asíncrona, minimizando el tiempo de sistema y maximizando la escalabilidad.

Principalmente se ha utilizado esta tecnología por el lenguaje utilizado para el backend, javascript. Generalmente, al realizar aplicaciones web, suele tener la ventaja de poder utilizar únicamente un sólo lenguaje para servidor y cliente.

Express

Express.js[2] es el entorno de trabajo para aplicaciones Node.js. Es de código abierto y tiene licencia MIT, siendo utilizado para la creación de webs o, en este caso, permite el guardado o la recuperación de datos de un usuario mediante una API.

Express funciona en la mayoría de paquetes de tecnologías al lado de Node.js y nos facilita mucho algunas funcionalidades como las rutas, especialmente importantes al desarrollar APIs.

MongoDB

MongoDB[3] es un sistema de bases de datos NOSQL, orientado a documentos y de código abierto cuyo esquema está basado en C++. Que la base de datos sea orientada a documentos nos proporciona una alta escalabilidad, así como un rendimiento bastante alto.

Esta base de datos ha sido escogida porque facilita mucho gran parte del trabajo. Los documentos, similares al formato de datos JSON, tienen una estructura muy similar a lo que se manipulará en Unity. Además, la base de datos no se centrará tanto en las relaciones entre usuarios, por ejemplo, teniendo únicamente peticiones sencillas como buscar, crear o actualizar los datos de un usuario.

Photoshop

Adobe Photoshop[8] es un editor de gráficos rasterizados creado por Adobe Systems. Su uso principal suele ser el retoque de fotografías y gráficos. Es, desde hace mucho, líder mundial en el mercado de la edición de imagenes, dominando el sector de una forma clara.

Photoshop ha sido escogido precisamente por eso, su uso tan extendido y la cantidad de recursos para su aprendizaje que se encuentran en internet han sido los motivos principales.Concretamente, me ha permitido crear tanto los iconos de los logros (ver sección 6.3.3) como el pseucódigo que se muestra de los pedidos (ver sección 6.2).

Blender

Blender[1] es un programa dedicado al modelado, renderizado,animación y creación de gráficos tridimensionales. Es multiplataforma y, posteriormente a su creación, pasó a ser software libre. Las nuevas versiones son compatibles con Windows, mac OS y Linux.

Esta herramienta se ha escogido por ser muy accesible y tener una gran cantidad de recursos de aprendizaje y de consulta disponibles. Además, su dominio en el sector y ser un programa conocido en todo el mercado lo hacen una buena elección. Blender me ha permitido crear y editar todos los modelos 3D que aparecen en el juego (ver sección 6.3.6).

Simulador de voz

En determinadas fases del juego, se ha decidido dar al jugador instrucciones oralmente. Al no tener acceso a los medios para grabar o conseguir un actor de voz, se ha utilizado una aplicación que permite simular sentencias. La aplicación se llama Narrator's Voice[4], y se encuentra disponible de forma gratuita para Android.

El simulador de voz ha sido escogido por su accesibilidad y por poder realizar tantas pruebas como hiciesen falta sin coste alguno. Además, tiene disponibles diversos idiomas, en cada uno de los cuales se puede escoger entre una voz femenina o masculina.

5 Análisis

En este apartado se enumeran primero los requisitos funcionales del proyecto. Después se describen cada uno de los casos de uno así como el respectivo diagrama (ver Figura 7).

5.1 Requisitos funcionales

Los requisitos funcionales del sistema serán:

- UC1: El usuario se registra en el juego.
- UC2: El usuario se autentica. Si ya está registrado, inicia sesión, recuperando así su progreso en el juego.
- UC3: El usuario inicia el juego.
- UC4: El usuario realiza el pedido (hamburguesa) para superar el nivel.
- UC5: El usuario sale del juego en cualquier momento (guardándose su progreso).
- UC6: El usuario pausa la partida.
- UC7: El usuario ve la lista de logros desbloqueados.
- UC8: El usuario corrige el último ingrediente que ha colocado.
- UC9: El usuario recibe un feedback después de cada entrega de un pedido.
- UC10: El usuario cambia el volumen de la musica o efectos.

5.2 Casos de uso

A continuación, se detalla las descripciones textuales de los principales casos de uso.

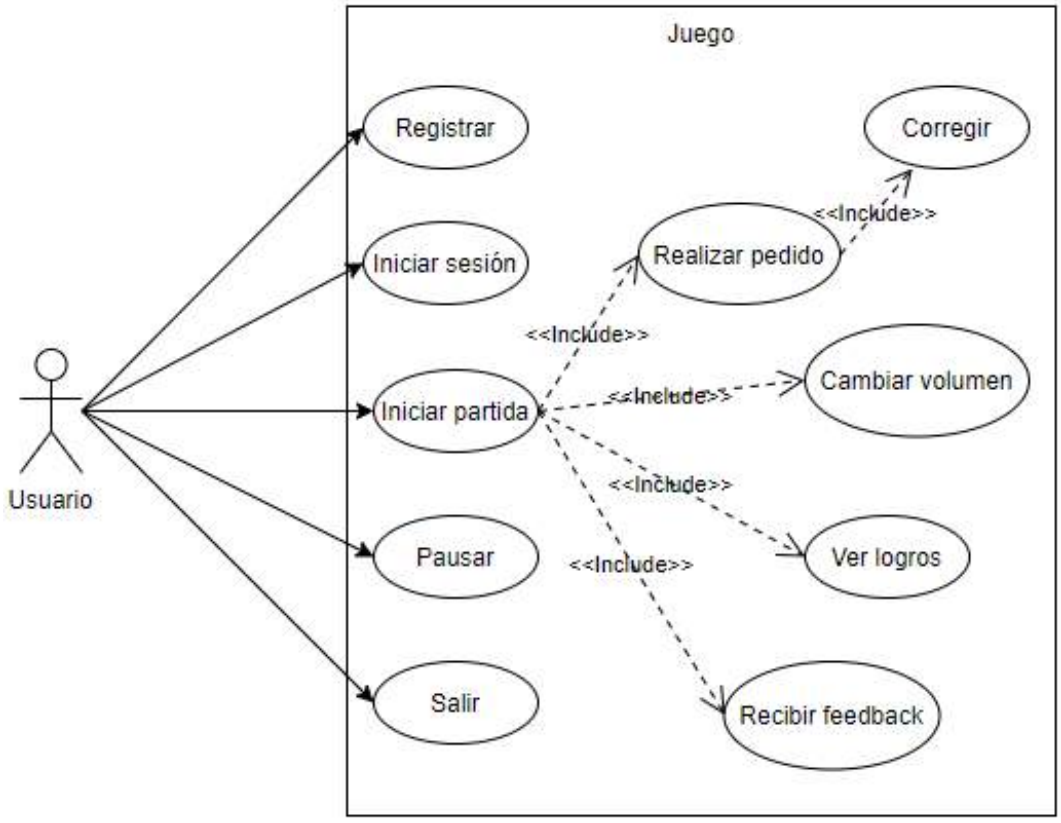

Figura 7: Diagrama de casos de uso (Fuente: propia)

UC1-Registrarse en el sistema

Descripción	El usuario desea registrarse como nuevo usuario en el juego
Precondición	Ninguna
Secuencia principal	1 El usuario inicia la aplicación 2 El sistema muestra el menú principal 3 El usuario selecciona la opción para registrarse 4 El sistema muestra el menú de registro 5 El sistema verifica que los datos introducidos no coinciden con otro usuario existente.
Flujo alternativo	5a Si los datos no son válidos, el sistema muestra de nuevo el menú de registro. 5b El sistema muestra de nuevo el menú de registro.
Post condición	El juego empieza con el usuario recién registrado.

UC2-Iniciar sesión en el sistema

Descripción	El usuario desea iniciar sesión en el juego
Precondición	Registrarse en el sistema (caso de uso 01)
Secuencia principal	1 El usuario inicia la aplicación 2 El sistema muestra el menú principal 3 El usuario selecciona la opción de iniciar sesión 4 El sistema muestra el menú de inicio de sesión 5 El sistema verifica que los datos introducidos coinciden con un usuario existente.
Flujo alternativo	5 Si los datos no son válidos, el sistema muestra de nuevo el menú de inicio de sesión.
Post condición	El juego empieza con el usuario correspondiente

UC3- Iniciar el juego

Descripción	El usuario inicia el juego en un nivel determinado
Precondición	El usuario se ha registrado (caso de uso UC1) o ha iniciado sesión (caso de uso UC2)
Secuencia principal	1 El usuario selecciona iniciar el juego 2 El sistema muestra el nivel.
Flujo alternativo	Ninguna
Post condición	El usuario puede avanzar niveles y jugar.

UC4- Salir del juego

Descripción	El usuario desea salir del juego con su progreso guardado
Precondición	Ninguna
Secuencia principal	1 El usuario pausa el juego. 2 El sistema muestra el menú de pausa. 3 El sistema guarda el progreso del usuario. 4 El sistema termina la partida.
Flujo alternativo	1a El usuario inicia la aplicación 2a El sistema muestra el menú principal 3a El usuario selecciona la opción de salir 4a El sistema termina la partida.
Post condición	La partida termina con el progreso del usuario guardado.

UC5- Pausar la partida

Descripción	El usuario desea pausar la partida
Precondición	La partida ha sido iniciada (caso de uso UC3)
Secuencia principal	1 El jugador solicita la pausa del juego 2 El sistema se pausa y muestra el menú de pausa
Flujo alternativo	Ninguna.
Post condición	El juego queda pausado en el menú de pausa hasta que se indique al sistema.

UC6- Ver la lista de logros

Descripción	El usuario desea ver la lista de los logros para conocer su progreso.
Precondición	La partida ha sido pausada (UC5)
Secuencia principal	1 El jugador selecciona la opción de logros en el menú de pausa 2 El sistema muestra el menú de logros.
Flujo alternativo	Ninguna.
Post condición	El usuario visualiza el sistema de logros hasta que desee reanudar la partida.

UC7- Jugar nivel (montar hamburguesa)

Descripción	El usuario desea montar una hamburguesa para completar un nivel.
Precondición	La partida ha sido iniciada (caso de uso 03)
Secuencia principal	1 El jugador realiza los movimientos necesarios para montar la hamburguesa. 2 El sistema comprueba que sigue quedando tiempo.
Flujo alternativo	2a Si el tiempo ya se ha consumido, el sistema saltará al día siguiente automáticamente.
Post condición	El usuario podrá ver el feedback al entregar la hamburguesa.

UC8- Corregir el último ingrediente

Descripción	El usuario desea eliminar el último ingrediente colocado en una hamburguesa.
Precondición	La hamburguesa está siendo montada (caso de uso UC7).
Secuencia principal	1 El jugador pulsa el botón correspondiente. 2 El sistema comprueba que exista un último ingrediente. 3 Una vez identificado, elimina el ingrediente. 4 El usuario visualiza la hamburguesa sin el ingrediente.
Flujo alternativo	2a En caso que no exista un último ingrediente, el sistema no hará nada.
Post condición	Ninguna.

UC9- Ver el feedback

Descripción	El usuario desea ver el resultado de la hamburguesa entregada.
Precondición	La hamburguesa ha sido montada (caso de uso UC7)
Secuencia principal	1 El jugador deposita la hamburguesa montada en la zona de entrega. 2 El sistema muestra un mensaje en función de si era la correcta o no.
Flujo alternativo	Ninguna.
Post condición	Se procede a la aparición de un nuevo pedido.

5.3 Grupos de usuarios

En el proyecto actual encontramos dos grupos de usuario. Por un lado los niños, de entre 12 y 16 años, que no han tenido contacto previo con la programación. Por otro, niños de la misma edad que tienen, más o menos, experiencia en el mundo de la programación.

5.3.1 Grupo de usuarios 1: jóvenes no iniciados en la programación

Este grupo de usuarios está formado por niños de entre 12 y 16 años. Tienen un conocimiento avanzado de tecnología, sabiendo controlar diferentes dispositivos y adaptándose muy bien al uso de nuevas plataformas. Suelen tener inquietudes y pensamientos muy diversos, desde temas sociales hasta más recreativos, como series o películas ...siendo un grupo bastante heterogéneo. Comparten, por otra parte, una atracción muy grande a nuevas formas de entretenimiento, encontrándose abiertos a cualquier novedad. En cuánto a su relación con la programación, es absolutamente nula, nunca han escuchado o visto ningún tipo de código, es decir, nunca han mostrado interés por la materia. Además, es un grupo poco relacionado con los juegos, habiendo jugado en ocasiones muy puntuales.

5.3.2 Grupo de usuarios 2: jóvenes iniciados en la programación

Este grupo de usuarios está formado por niños de entre 12 y 16 años. Tienen un uso de la tecnología muy avanzado, conociendo y controlando el funcionamiento de los dispositivos que utilizan. Generalmente, tienen inquietudes relacionadas con el apartado más tecnológico, con el mundo de los videojuegos y su desarrollo, siendo un grupo bastante más enfocado en el dominio a tratar. Suelen estar bien informados sobre las nuevas tecnologías, conociendo ya seguramente las gafas de realidad virtual más en profundidad. En cuánto a su relación con la programación, es bastante cercana, habiendo escuchado o incluso probando algunos códigos o ejercicios, es decir, muestran un interés creciente por la materia. A parte, es un grupo muy relacionado con el mundo de los videojuegos, sabiéndose adaptar y abriéndose rápido a nuevas mecánicas.

5.4 Personas

Una vez se han definido los grupos de usuarios, se crean personajes ficticios llamados Personas, construidos a partir de una descripción y una imagen. Las Personas, permiten tener al usuario en cuenta a lo largo de todo el proceso de diseño. Cada uno de estos personajes ficticios se crea a partir de un grupo de usuarios, permitiendo unir todas sus características y motivaciones bajo un nombre concreto.

Joel

Joel es un chico de 13 años que se encuentra cursando 2n de la ESO en Hospitalet de Llobregat. Durante su tiempo libre, le gusta salir con sus amigos. Suele leer mucho, utilizando para ello un libro físico, su móvil, la tablet o el ebook, dispositivos que domina con mucha facilidad. Además, está pendiente del mercado de portátiles, buscando siempre una oportunidad para poder comprar uno. Su objetivo es llegar a ser profesor, aunque también le gusta mucho tocar la guitarra. Joel suele tener inquietudes relacionadas con temas sociales, teniendo mucha afinidad con la causa animalista.

Figura 8: Foto de Joel (Fuente: `https://bit.ly/32nR1jI`)

Raquel

Raquel es una chica de 14 años que estudia tercero de la ESO en Barcelona. Durante su tiempo libre, siempre está buscando juegos en oferta en las diferentes plataformas digitales. Suele jugar juegos cooperativos, aunque a veces le apetece jugar sola. Suele tener inquietudes relacionadas con el mundo de la tecnología, siempre dispuesta a estar a la moda de los nuevos dispositivos que salgan al mercado. Además, conoce y le inquieta el funcionamiento interno de los juegos, y ha empezado algunos proyectos con Scratch para programar pequeñas aplicaciones de forma autónoma. Su objetivo es llegar a ser ingeniera para ser CEO de una gran empresa de juegos y además pasárselo bien con sus amigos.

Figura 9: Foto de Raquel (Fuente: `https://bit.ly/32rOAfK`)

6 Diseño del juego

6.1 Canvas de diseño

A continuación, se presenta un canvas sobre el diseño del juego (Game Design Canvas). La intención de este canvas es ayudar a organizar el proceso de diseño conceptual del juego, así como analizar posibles puntos débiles del diseño. Los elementos que encontraremos serán los siguientes:

- Planificación: El primer paso será analizar de qué recursos disponemos para el desarrollo del juego. Se ha de definir: el género, el presupuesto, el plazo para desarrollar el juego, los recursos de personal, los usuarios a los que va dirigido y la plataforma sobre la que funcionará.
- Meta de diseño: Las emociones que se evocarán en los jugadores al jugar. En este caso, la diversión y la curiosidad por el aprendizaje sobre programación.
- (1) Deseos/Necesidades: El juego permite a los jugadores satisfacer algunos deseos o necesidades producto de su condición humana. En este punto, se recoge una lista de dichos deseos y necesidades. En el presente proyecto, el usuario busca introducirse en la programación de una forma sencilla y divertida. Puede resolver problemas y sentirse realizado por ello.
- (2) Objetivos: Existen diferentes tipos:
  - De corto alcance: Actividad cíclica central. El diseño de aquella acción, o conjunto de acciones, que los jugadores deben realizar durante una sesión de juego. En este caso, analizar y realizar el pedido de forma correcta.
  - De medio alcance: Elementos que los jugadores acumularán en preparación para objetivos de largo alcance. Son objetivos intermedios, logrados durante varias sesiones de juego. Por ejemplo, acabar el día con la mejor puntuación posible para lograr un determinado logro propuesto.
  - De largo alcance: El objetivo o desafío final que enfrentarán los jugadores, tras el cual el juego habrá concluido. En *Cooking Code* será completar todos los logros.
- (3) Sistema de progresión: El sistema requiere de mecanismos que hagan sentir al usuario que avanza y progresa. Los jugadores percibirán dicho avance a través de este sistema. Es uno de los mecanismos cruciales de fidelización. En este proyecto el personaje jugable no evolucionará. Será la habilidad del propio jugador, así como su ventana de logros, que irá cambiando a lo largo del juego. Además, cada vez habrá más ingredientes y posibilidades, materializando esa progresión en los grandes platos que se realizarán.
- (4) Desafíos: Los obstáculos que experimentan los jugadores deben de estar siempre un poco por encima de su habilidad actual. El diseño debe incluir una curva de dificultad ascendente, gradual, con períodos de calma dónde se pueda hacer patente el progreso realizado. En el juego, la curva de dificultad se basará en la inclusión de nuevos ingredientes e instrucciones, así como un incremento de la complejidad de los pedidos. Estos cambios serán graduales, nunca incluyendo más de uno por nivel.
- (5) Sistema de recompensas: A través de este, los jugadores pueden visibilizar su progreso. Los elementos del sistema suelen ser insignias, logros, títulos honoríficos o cualquier elemento coleccionable. En *Cooking Code* las recompensas podrán ser inmediatas, como por ejemplo, los comentarios de los clientes.Completando los pedidos con éxito el jugador podrá recibir medallas correspondientes a los logros completados, visualizando su progresión siempre que quiera.
- Narrativa: Si se dota de una estructura dramática al juego se puede conseguir un efecto de retención o de motivación mucho mayor. En dicho apartado, se describe el contexto en el que sucede la experiencia diseñada. En este caso, el juego se ambienta en un contexto futurista donde la sociedad ha evolucionado. Ahora, la mayoría de las lenguas se han mezclado con los lenguajes de programación, producto de una gran revolución tecnológica. Por ello, los pedidos de la hamburguesería llegan en forma de esa extraña mezcla: el pseudocódigo.

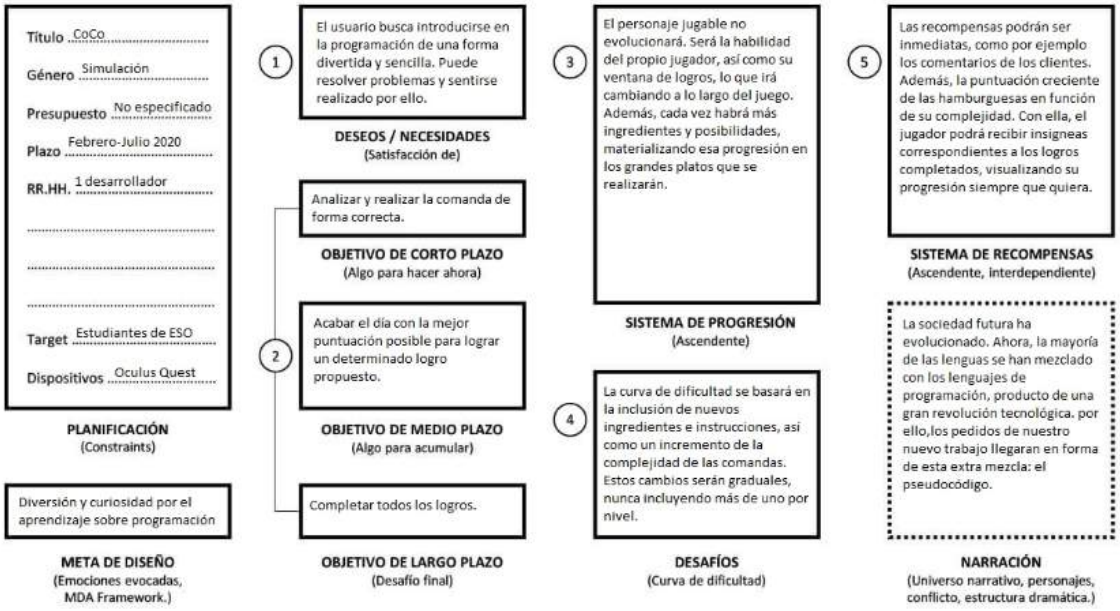

Figura 10: Canvas de diseño (Fuente: propia)

6.2 Pedidos

El diseño de los pedidos (ver Figura 11) está basado en la programación visual existente en plataformas como Scratch o CoSpaces. Cada una de las acciones a realizar, o instrucciones, viene representada por un bloque. El pedido, al final, es un conjunto de bloques que encajan entre si en un orden determinado [1].

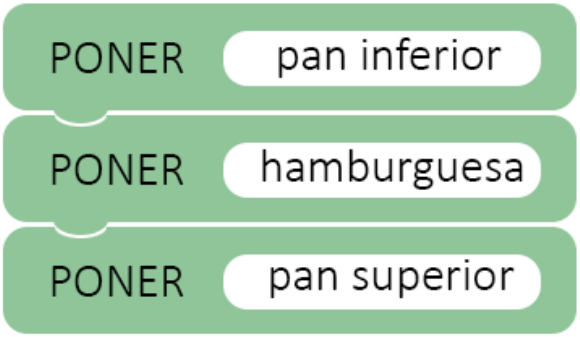

Figura 11: Primer pedido básico (Fuente: propia)

[1] Ver vídeo de resolución de los pedidos básicos: `https://www.youtube.com/watch?v=aV7KA7_Eigs`

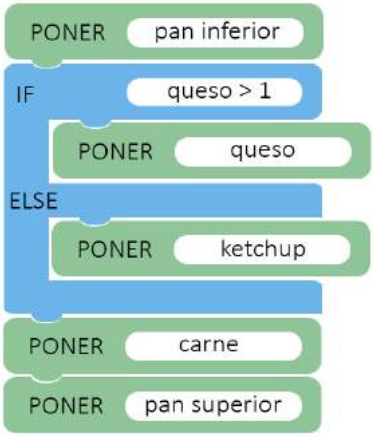

Figura 12: Pedido condicional (Fuente: propia)

Como resultado final de realizar cada una de las acciones, el usuario tendrá que ser capaz de montar la hamburguesa deseada. Cada uno de los bloques viene dividido en dos partes bien diferenciadas:

- Acción a realizar: Generalmente será la acción "PONER", ya que es la que se ha implementado en esta iteración del proyecto.
- Ingrediente: En un rectángulo blanco, a la derecha del bloque, se especifica sobre qué ingrediente se ha de realizar la acción. Generalmente, indicará el siguiente ingrediente a poner en la hamburguesa.

No obstante, esto es únicamente aplicable a los pedidos más básicos. Existen 2 tipos de pedidos más donde aparecen bloques de una forma y con una lectura diferente, éstos se desarrollan en las siguientes secciones.

6.2.1 Instrucciones condicionales

En las instrucciones condicionales (ver Figura 12) encontraremos el bloque condicional[2]. Este nuevo bloque estará dividido en dos partes:

- IF: Es la primera parte del bloque, en ella se encuentra la condición que se ha de cumplir. En caso de ser cierta, se ejecutará el bloque que contiene justo debajo.
- ELSE: Es la segunda parte del bloque, en ella se encuentra contenido el bloque que tendremos que ejecutar en caso de que la condición mostrada al inicio no se cumpla.

[2]Ver vídeo de realización de los pedidos con instrucciones condicionales: `https://www.youtube.com/watch?v=4I8x0JyHFNM`

6.2.2 Instrucciones iterativas

En las instrucciones iterativas(ver Figura 13) encontraremos el bloque FOR. Este nuevo bloque contendrá en su interior uno o más bloques básicos. Además, en el se indicará cuantas veces se han de repetir dichos bloques[3].

Con la inclusión de los bloques de esta forma, se deja muy clara la jerarquía, y no existe ambigüedad sobre que bloques entran o no en el bucle a realizar.

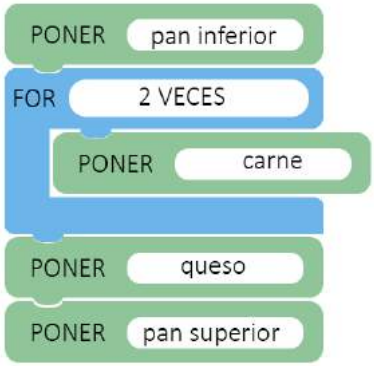

Figura 13: Pedido iterativo (Fuente: propia)

6.3 Jugabilidad y mecánicas

El usuario tiene el rol de cocinero en una hamburguesería, teniendo que realizar todos los pedidos que van llegando. A su disposición tendrá un conjunto de ingredientes, los cuales no son infinitos, sino que tienen una cardinalidad que indica su disponibilidad.

El juego se organiza en jornadas de trabajo, que permiten reiniciar las cardinalidades de los ingredientes, así como mostrar la experiencia conseguida. El objetivo del jugador será, para cada jornada, hacer todas las hamburguesas que pueda. El jugador podrá saber en que momento del día se encuentra, ya que hay un reloj en la escena que marca cada hora[4].

Primero, el usuario verá en la pantalla el pedido que ha de realizar. Una vez comprenda el pseudocódigo, tendrá que empezar a realizar el montaje de la hamburguesa. No hay tiempo límite por pedido, únicamente existe el limitante del final de día. Algunos ingredientes, requieren de acciones específicas, por ejemplo, la carne de la hamburguesa tiene que ser cocinada antes.

La interacción principal será el agarre. El jugador cogerá un ingrediente de su cesta y lo mantendrá agarrado, dejándolo en la zona remarcada en el plato. De está forma, la hamburguesa irá tomando forma. Una vez finalizada, el jugador cogerá el plato de nuevo, depositándolo en la zona de entrega, justo delante suyo.

Como se ha mencionado anteriormente, los ingredientes no son infinitos, tienen una cardinalidad. Con tal de conseguir dichos ingredientes, tendremos que cogerlos del interior de su correspondiente recipiente.

[3]Ver vídeo de resolución de los pedidos iterativos: `https://www.youtube.com/watch?v=m_uSVEkhEBQ`
[4]Ver vídeo para ver cómo pasan los días: `https://www.youtube.com/watch?v=t_7n_6BZ--w`

Cada vez que saquemos uno, se descontará del contador que aparece justo delante. De la misma forma, si retornamos un ingrediente a su recipiente, sumará uno a su cardinalidad.

Los principales beneficios de estas mecánicas e interacciones es su similitud con la realidad. El juego intentará que las tareas y los movimientos necesarios para cumplir los objetivos del juego sean lo más similares posibles a una cocina real. De esta forma, utilizamos una metáfora que permite una familiarización mucho más rápida y divertida del usuario, además de proporcionar más comodidad en las interacciones con el entorno.

6.3.1 Curva de aprendizaje

El juego contendrá una curva de aprendizaje simple[5]. Los primeros pedidos que realice el usuario serán los más básicos, con pocos ingredientes y con cardinalidades muy grandes. De está forma, se pretende que la introducción sea lo más suave posible, haciendo que sea gratificante y el jugador perciba cierto progreso.

Una vez se hayan logrado cierto numero de pedidos básicos con éxito, se empezarán a introducir aquellos de una complejidad mayor (condicionales e iterativos), así como nuevos ingredientes.

Cabe destacar que, con la inclusión de estos nuevos pedidos, no desaparecerán los más básicos, simplemente aparecerán de forma aleatoria unos u otros. De está forma, se quiere evitar que el juego sea excesivamente lineal o monótono cuando el usuario no es capaz de realizar un tipo de pedido.

Cada vez que un tipo de pedido nuevo sea desbloqueado, se le hará saber al usuario. Además, irá acompañada de un pequeño tutorial para aprender a abordar el nuevo tipo, el cual podrá ser consultado posteriormente por el usuario cuando quiera.

De está forma, se espera que el usuario pueda ir experimentando una curva de aprendizaje siempre creciente pero muy suave y constante, poniendo a su disposición en todo momento recursos que le permitan repasar algunos conceptos.

6.3.2 Feedback del pedido

Cada vez que el jugador entrega un pedido éste recibe un feedback en forma de mensaje. El diseño y la elección de estos mensajes es muy importante. Por una parte, se ha de intentar que sean lo más claros y concisos posible, sin ambigüedades, de tal forma que se pueda aprender de ellos. Por otro lado, la forma es muy importante, y más tratándose del perfil de usuarios con el que estamos tratando. Se ha buscado mostrar mensajes de ánimo independientemente del resultado, palabras sencillas, sin buscar un estilo excesivamente formal[6].

Dar un buen feedback es muy importante. No sólo permite al usuario conocer su error y evitar futuras frustraciones, sino que puede llegar a alentarle o hacerle mucho más amena la experiencia. Además, en juegos donde el aspecto formativo es tan importante, su claridad es clave.

[5] Ver vídeo tutorial: `https://www.youtube.com/watch?v=tPMWkB3xhsw`
[6] Ver vídeo sobre el feedback: `https://www.youtube.com/watch?v=749mQYmebPQ`

6.3.3 Sistema de logros

Es muy importante mantener la motivación y la sensación de progreso presente en el usuario. Por eso, se ha decidido incluir un sistema de logros, basados en el número de pedidos realizados con éxito por el usuario. De cada uno de los tres tipos de pedidos disponibles (básicos, condicionales, iterativos) existirán tres logros para desbloquear [7]:

- Primer logro: El primer logro (ver Figura 14) se desbloqueará al realizar el primer pedido de ese tipo de forma correcta. Por ejemplo, la primera vez que realicemos correctamente un pedido condicional se desbloqueará dicho logro.

Figura 14: Logro al realizar un pedido condicional (Fuente: propia)

- Segundo logro: El segundo (ver Figura 15) será desbloqueado a los 10 pedidos, recompensando el progreso, evidenciando que se empieza a dominar ese tipo de pedidos.

Figura 15: Logro al realizar 10 pedidos condicionales (Fuente: propia)

- Tercer logro: El último logro (ver Figura 16) será desbloqueado a los 40 pedidos (para los básicos) o a los 30 (para los demás), recompensando así al jugador por el dominio de ese tipo.

[7] Ver vídeo sobre los menús y los logros:`https://www.youtube.com/watch?v=djhe1fxDlWo`

Figura 16: Logro al realizar treinta pedidos condicionales (Fuente: propia)

Además, el incluir el sistema de logros permite ver el progreso del usuario de una forma más clara. Así, si el juego se utilizase en un aula, el profesor podría ver qué está resultando más complicado para un alumno en concreto o cuál es su grado de dominio.

6.3.4 Control de errores: Undo

Cuando el juego está siendo diseñado, se ha de pensar mucho en que funcionalidades pueden o no beneficiar al usuario. En determinados juegos, la posibilidad de deshacer el último paso realizado es absurda, o suprime el posible reto que suponía el nivel.

No obstante, el objetivo principal del proyecto no es realizar un juego normal, sino uno con un componente formativo claro. Por ello, ésta funcionalidad toma mucho sentido y tiene mucha utilidad para el usuario. Se pretende así reducir el estrés o la frustración producida cuando un error involuntario ocurre. Permitiendo también corregir errores voluntarios si el usuario se da cuenta antes de la entrega[8].

De esta forma, el usuario se centra mucho más en comprender y entender el pedido, sabiendo que cualquier error o confusión pueden ser corregidos con rapidez y facilidad.

6.3.5 Distribución de la mesa de pedidos

La distribución de la mesa donde "trabajará" el jugador es muy importante. Se pretende tratar de evitar movimientos innecesarios, bruscos o que puedan causar confusión.

Es por eso que el diseño agrupa los elementos en función de su posición y el orden en el que serán escogidos. En un lado de la mesa, encontramos todos los ingredientes excepto la carne. Los más cercanos a la zona de trabajo son los más comunes, el pan superior y el pan inferior. Con ello, conseguimos que el desplazamiento la mayoría de veces sea mínimo. En el otro lado, la carne se encuentra justo al lado de la plancha, de esta forma, el movimiento para cocinarla es sencillo e intuitivo, mucho más organizado.En el centro, encontramos los platos y el ketchup, los cuales suelen ir al inicio o al final de los pedidos.

La zona de entrega ha tenido diferentes localizaciones durante el proceso de diseño. Primero, pensé en colocarla en uno de los laterales, con el objetivo de hacer la zona de trabajo mucho más limpia y amplia. Sin embargo, el tener que estar constantemente girando la cabeza, y perder de vista el pedido, me parecía algo negativo. También pensé en colocarla en una zona elevada, no obstante, el movimiento es

[8]Ver vídeo sobre el Undo: `https://www.youtube.com/watch?v=22A5mL2e6MA`

mucho menos natural y los beneficios no son tantos como los inconvenientes. Finalmente, dicha zona será colocada justo delante de la zona de montaje. De esta forma, se mantiene todo el rato contacto directo con el pedido, el movimiento en realidad virtual es natural y el jugador puede comprobar rápidamente el feedback que aparece.

6.3.6 Colores y modelos

Todos los modelos que aparecen a lo largo del juego han sido modelados por mí, lo cual me permitía dotarlos de un estilo uniforme y característico. El estilo escogido ha sido *Low poly*, ya que estéticamente es bonito, no es muy complejo de modelar y además le da un toque más inmaduro y divertido al juego. Uno de los elementos de la escena del juego que más destaca es la televisión, dónde el jugador estará mirando la mayoría del tiempo. Con el objetivo de hacerlo más familiar, he decidido personificarla, dándole una vestimenta, un nombre y referencias al mundo de la cocina, como puede ser el gorro. Todas estas decisiones de los modelos han sido pensadas para hacer el entorno agradable y relajado, atractivo para el usuario. Por otra parte, los colores vienen a reforzar también esta idea. No se ha querido utilizar colores con una intensidad muy alta, ni contrastes que pudiesen ser molestos al verlos durante mucho tiempo. La mayoría de materiales son suaves, uniformes, y todos conforman una paleta de colores muy agradable. Además, se han buscado colores bastante neutros, que fuesen del agrado de todo tipo de jugadores.

6.4 Música

La música es un componente diferencial en un juego. No obstante, al no tener ningún tipo de formación musical, he decidido seleccionar música sin copyright disponible en plataformas digitales. La música no tendrá un papel principal, simplemente estará de fondo, pudiéndose desactivar por completo des del menú de pausa. Es una melodía bastante relacionada con el mundo de la cocina, muy calmada, aunque con un ritmo bastante entretenido que refuerza el concepto que quería expresar con los colores y modelos[11].

7 Diseño de la aplicación

En este apartado se presenta el diseño del software que se ha desarrollado para crear el juego. Primero, se presenta un diagrama de clases completo, detallando después las clases más importantes y su funcionalidad. Después, se detalla el flujo de ejecución (diagrama de flujo) en los menús o en la realización de un pedido. A continuación, se explicará el uso principal de la base de datos, así como la estructura de su contenido. Se describen también los principales patrones de diseños, justificando su uso y exponiendo sus beneficios. Finalmente, se enumeran los requisitos generales para poder desarrollar el juego.

7.1 Diagrama de clases

A continuación se especifica el diagrama de clases (ver Figura 17), incluyendo únicamente aquellas clases que han sido creadas. A parte, se han debido de codificar y modificar determinadas funcionalidades de los paquetes proporcionados por Unity.

A continuación se describirán brevemente las clases más importantes.

Figura 17: Diagrama de clases del proyecto (Fuente: propia)

7.1.1 Controlador de partida

La clase *controladorPartida* (ver Figura 18) se encuentra conectada con una gran cantidad de clases. Esto es debido a que es la encargada de gestionar el bucle principal del juego. Es decir, será en está clase donde encontremos el lanzamiento de todos los eventos, la gestión de la pausa del juego, etc... . Como se puede ver, se encuentra también en contacto directo con otros controladores, que permiten acceder a determinadas partes del código. Por ejemplo, en vez de usar directamente a las instancias de los logros, accedemos a *LogrosController* como único punto de acceso, haciéndolo todo mucho más ordenado. Además, hay una única instancia ya que esta clase utiliza el patrón de diseño Singleton (ver sección 7.4.2),

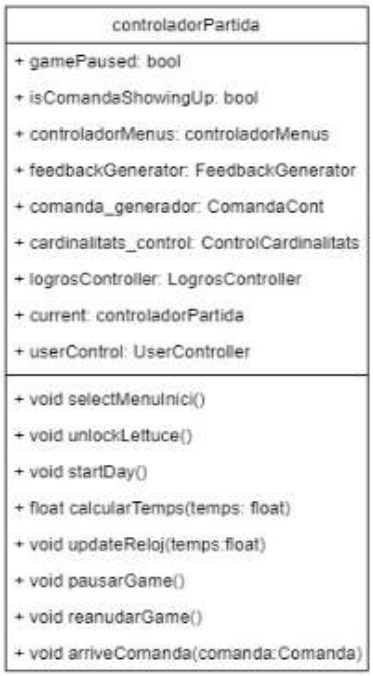

Figura 18: Clase controladora de la partida (Fuente: propia)

7.1.2 Undo

La clase *UndoManager* (ver Figura 19) es la encargada de gestionar todo el sistema de corrección. Podemos ver como se encuentra instanciada en *AttIngredients*, clase asociada a los platos que se encarga de ensamblar los ingredientes una vez están en ellos. De esta forma, cuando el evento se dispara, la clase *UndoManager* permitirá saber si el objeto a eliminar está o no en el plato. Además, podemos ver que tiene una relación directa con la clase *Comanda*, que mantendrá la información de qué ingredientes incluye la hamburguesa que está siendo montada. De está forma, no sólo el ingrediente se eliminará del plato, sino también de la lista de ingredientes actuales.

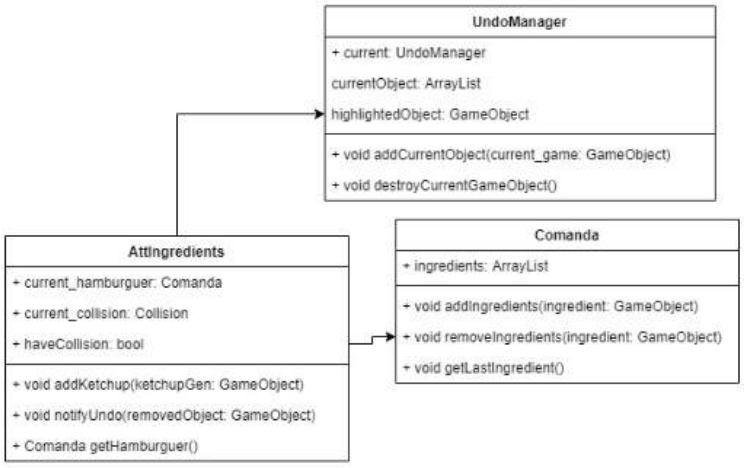

Figura 19: Clase controladora de la funcionalidad Undo

7.1.3 User

La información del usuario nos llega desde el backend en formato JSON, siendo transformada en un conjunto de clases (ver Figura 20). Estas clases son utilizadas para volcar la información, teniendo el decorador de serializable y exactamente los mismos argumentos que el JSON. Para mantener el acceso ordenado y sin problemas, está la clase *UserController*, que es a la que consultará constantemente el controlador de partida (ver sección 7.1.1). Esta clase únicamente tiene esa utilidad, proporcionar los métodos de consulta de información de *UserJSON*, *DataLevel* y *LogrosJSON*.

7.1.4 Controlador de menús

El controlador de menús (ver Figura 21) es la clase que interacciona con toda la interfaz gráfica de la aplicación. Como se puede ver en los métodos que contiene, es la única vía de acceso para activar o desactivar los diferentes menús de la aplicación. Esto, nos permite mantener un orden y saber en todo momento dónde se está activando, cambiando o desactivando una determinada ventana. Como se puede ver, también contiene una instancia de ella misma, aplicando el patrón Singleton (ver sección 7.4.2).

7.1.5 Pedidos

Los pedidos, al igual que la información del usuario, viene originalmente en un formato JSON, aunque en este caso está guardado localmente. La clase *ComandaJSON* (ver Figura 22) representa cada pedido leído del JSON, siendo todos almacenados en la clase *JSONReader*. Está clase es la que contendrá el fichero original, además de ser la encargada de desbloquear los nuevos niveles y generar los pedidos. Al igual que hemos visto anteriormente, para evitar tener accesos no necesarios a estas clases, tenemos un punto de acceso llamado *ComandaCont*. La clase *ComandaCont* es la que se encarga de gestionarlo todo, mientras que las dos anteriores tenían una funcionalidad más de contenedor. Será la clase que llamará el controlador de partida cuando toque generar un pedido nuevo o quiera conocer el estado del actual.

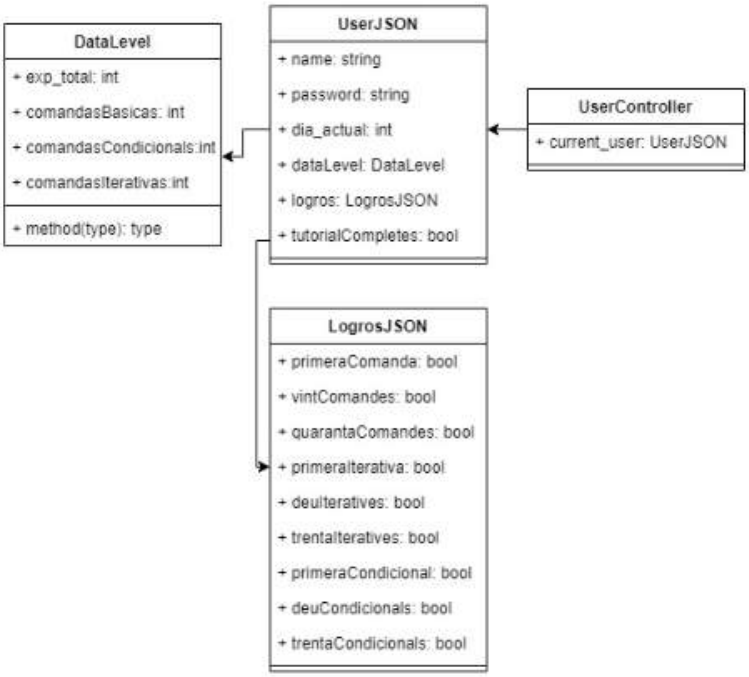

Figura 20: Clases referente al usuario (Fuente: propia)

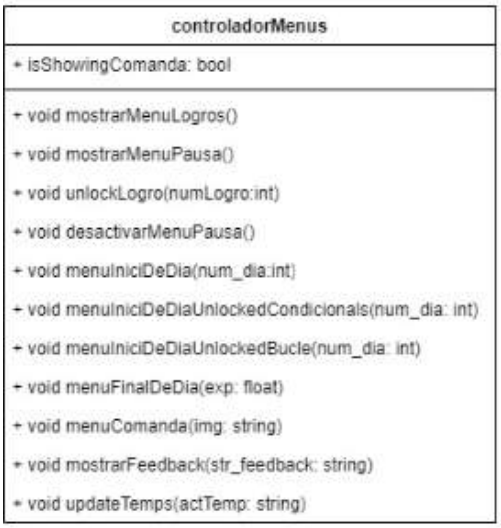

Figura 21: Clases controlador de los menús (Fuente: propia)

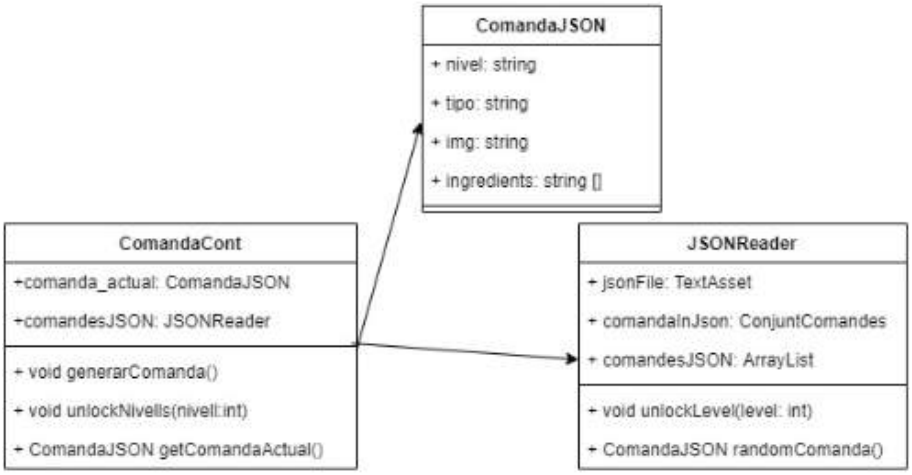

Figura 22: Clases referentes a los pedidos (Fuente: propia)

7.1.6 Creación de ingredientes

Para una correcta creación de los ingredientes en la escena se ha decidido utilizar el patrón de diseño Factory (ver sección 7.4.1). La clase abstracta *IngredientCreacio* (ver Figura 23) será la clase de la cual extienden todas las clases de creación, obligando así a la implementación del método para crear el objeto correspondiente. Por otro lado, la que será llamada y visible para el resto del proyecto será *FactoryIngredients*.

7.1.7 Feedback

La clase que generará el feedback (ver Figura 24) que se mostrará cuando se entregue un pedido será *FeedbackGenerator*. Su único atributo es el pedido que se está mostrando actualmente en la pantalla, recibiéndolo de *ComandaCont* (ver sección 7.1.5). Además, el algoritmo encargado de generarlo retorna una tupla (string,float), donde por una parte se pasa el mensaje a mostrar y por la otra la puntuación asociada.

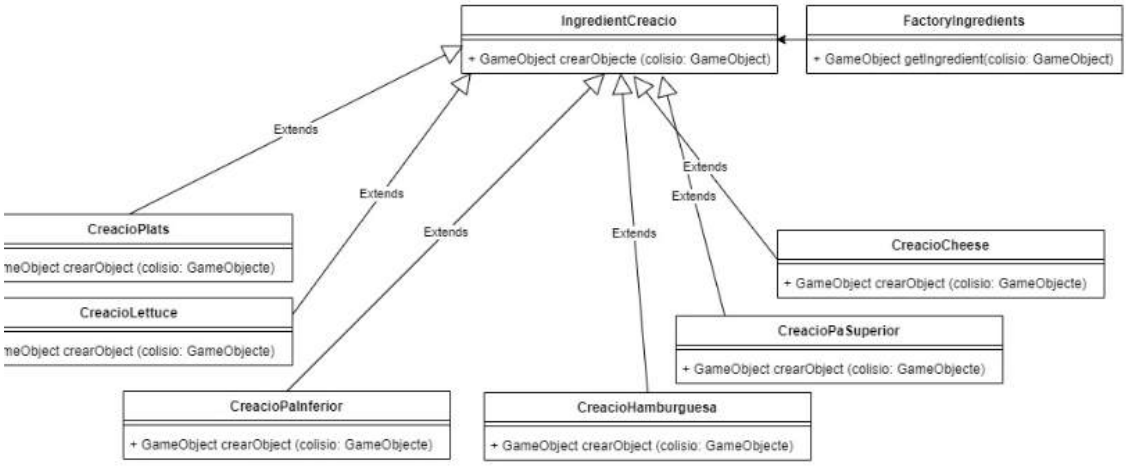

Figura 23: Clases referentes a la creación de ingredientes (Fuente: propia)

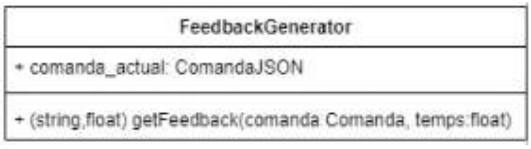

Figura 24: Clase generadora del feedback del pedido (Fuente: propia)

7.2 Diagrama de flujo

En este apartado se adjuntarán los diagramas de flujo que permitirán entender la lógica de cada una de las partes del juego[9][10].

Figura 25: Inicio de partida (Fuente: propia)

[9]Ver vídeo de registro: `https://www.youtube.com/watch?v=FQtjIP4aHRA`
[10]Ver vídeo inicio de sesión: `https://www.youtube.com/watch?v=djhe1fxDlWo`

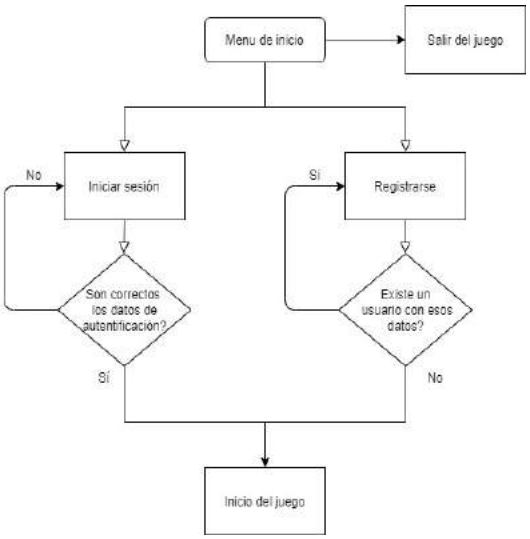

Figura 26: Menú de inicio (Fuente: propia)

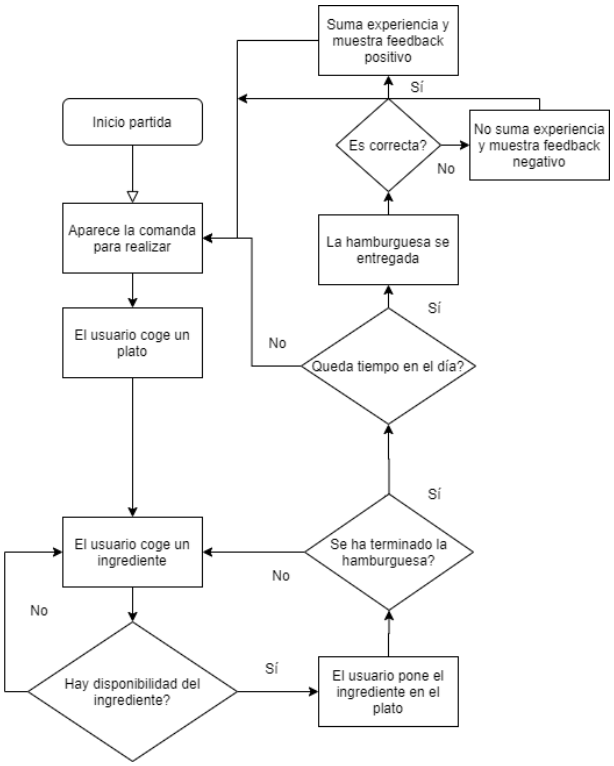

Figura 27: Desarrollo de la partida (Fuente: propia)

7.3 Estructura de la base de datos

La base de datos tendrá la funcionalidad de guardar toda la información referente al progreso y las preferencias del usuario. De esta forma, varias personas podrían utilizar la aplicación en el mismo dispositivo, cada uno iniciando sesión previamente con su cuenta. En esta base de datos, vamos a querer guardar información necesaria para la autenticación del usuario, información sobre su progreso y preferencias como el volumen de la música y si ha terminado o no el tutorial. La base de datos utilizada es no relacional, basada en documentos. Estos documentos tienen que tener un esquema fijo, encontramos tres tipos de esquemas en la base de datos:

El esquema principal, corresponde al jugador. Primero, encontramos los campos necesarios para la autenticación, y otros como el día en el que se ha quedado o si ha completado o no el tutorial. Esquema del usuario (*UsuariSchema*):

- name : string / Obligatorio
- password: string / Obligatorio
- diaActual : Number / Default: 1
- dataLevel: dataLevelSchema
- logros: logrosSchema
- tutorialCompletes : Boolean / Default: false
- volumen: string / Default: "1.0"

Por otro lado, tenemos el esquema de datos de la partida, que guarda la experiencia acumulada del usuario y el número de cada tipo de pedidos que ha completado con éxito.

Esquema de datos de la partida (*dataLevelSchema*):

- expTotal : Number
- comandasBasicas : Number
- comandasCondicionals : Number
- comandasIterativas : Number

Por último, el esquema de logros, que guarda si el usuario ha completado el logro correspondiente. De esta forma, guardamos toda la información relativa al usuario y a su progreso. Esquema de logros (*logrosSchema*):

- primeraComanda : Boolean / Default: false
- vintComandes : Boolean / Default: false
- quarantaComandes : Boolean / Default: false

- primeraIterativa : Boolean / Default: false
- deuIteratives : Boolean / Default: false
- trentaIteratives : Boolean / Default: false
- primeraCondicional : Boolean / Default: false
- deuCondicionals : Boolean / Default: false
- trentaCondicionals : Boolean / Default: false

7.4 Patrones de diseño

En este aparto se explicará como se ha estructurado el código y los patrones de diseño que se han utilizado.

7.4.1 Factory

El patrón Factory es un patrón de diseño creacional, que permite construir una jerarquía de clases. La principal ventaja es que se oculta al resto de clases los detalles de la creación de dichos objetos.

El contexto bajo el que este patrón se ha utilizado es el siguiente: Cuando se trata de coger un nuevo ingrediente, tenemos los diferentes tipos de generados dispuestos en la escena. En función del generador que seleccionemos, obtendremos un ingrediente u otro. De está forma, se tendrá que cambiar el comportamiento de la clase *OVRGrabbable*, que modula el comportamiento de la mano al tocar y coger un objeto. Está clase, por otro lado, viene incluida dentro del paquete de soporte para Oculus Quest de Unity, por lo tanto, contiene una gran cantidad de código y no queremos tener mezclado el propio con otro ya escrito. De está forma, cuando se llame al evento de dicha clase producto de tocar un objeto, comprobaremos si es un generador y, en caso de serlo, instanciaremos el objeto necesario. No obstante, no queremos que todo ese código sea visible para el resto de clases y quede incrustado en otra que es ya de por si muy grande. Además, si queremos cambiar o incluir un comportamiento nuevo en esta instanciación deberíamos de tocar una clase proporcionada por Unity, lo cuál aumenta la probabilidad de producir un error de forma involuntaria.

- *IngredientCreacio*: Es la interfaz abstracta que compartirán todos los ingredientes. De está forma, nos aseguramos que todas las clases que correspondan a un ingrediente tengan el método que permita instanciar el objeto correspondiente.
- *FactoryIngredients*: Es la clase que será visible para el resto de clases del proyecto. En ella, en función del nombre del generador seleccionado, se llamará a la creación de un ingrediente u otro. Aún así, sólo permite escoger la clase creadora, para ella todo el procesos sigue siendo invisible.
- *CreacioIngredient*: Conjunto de clases, cada una con el nombre de su ingrediente, que heredan de IngredientCreacio y que son instanciadas en FactoryIngredients. Son ellas las que contienen la instanciación y posicionamiento del nuevo objeto.

De esta forma, si quisiéramos añadir un nuevo generador, lo único que tendríamos que hacer sería añadir una nueva entrada en *FactoryIngredients* y crear la clase hija de *IngredientCreacio* correspondiente. Se

puede ver como en ningún momento la clase *OVRGrabbable* o el resto del proyecto cambian. Para ellos el funcionamiento y el código sigue siendo el mismo, encapsulando completamente la creación y evitando problemas a la hora de escalar el numero de ingredientes.

7.4.2 Singleton

Singleton es un patrón de diseño que nos permite restringir la creación de objetos de una determinada clase, asegurándonos de está forma que siempre trabajamos con una misma instancia. Este patrón ha sido utilizado en dos clases, cada una de ellas por un motivo distinto:

- *ControladorPartida*: Como su nombre indica, es la clase que controlará todas las variables y orquestará los diferentes eventos o comunicaciones que se realicen en el juego. Es quién contendrá todas las variables que determinan el estado de la partida, instanciando o creando eventos cuando sea necesario. Para esta clase, nos interesa tener una única instancia, que además será accesible para una gran cantidad de objetos. Al ser el estado del juego un recurso disponible para toda la aplicación, aplicar el patrón Singleton facilitaba mucho el control, asegurándonos que siempre estamos trabajando con los mismos datos.

- *ControladorMenus*: Esta clase controlará el acceso a todos los menús, o datos que se vayan a mostrar en pantalla. De esta forma, tenemos un punto de acceso único a toda la interfaz gráfica, evitando posibles conflictos. El patrón Singleton aquí nos asegura que existe ese único punto de entrada hacía los menús, asegurándonos siempre de qué una sola instancia este mostrando un determinado mensaje o menú.

7.4.3 MVC

El patrón modelo-vista-controlador es un patrón de arquitectura de software, que nos permite organizar y separar los datos de una forma más efectiva. De está forma, conseguimos separar cada una de las responsabilidades, permitiéndonos un mejor desarrollo y una escalabilidad más fácil.

En este caso, el patrón ha sido utilizado para el backend, organizando las diferentes clases en:

- Modelo: Es donde tendremos las clases que conectan con la base de datos, así como los esquemas que definirán como serán dichos datos.

- Controlador: En este caso, el controlador contiene gran parte de la lógica, y nos permite comunicar de forma segura la vista con el modelo.

- Vista (Rutas): Tendremos la clase que definirá las diferentes rutas de nuestra aplicación. Además, definiremos que método del controlador tratará la petición que se realice a la dirección web.

De esta forma, el código queda mucho más legible y sencillo de utilizar. Por ejemplo, si queremos añadir una ruta nueva, no tenemos porqué escribir código en las clases encargadas de conectar la base de datos, por ejemplo, haciendo mucho más seguro el desarrollo.

7.5 Requisitos generales para el desarrollo

7.5.1 Software

En este apartado se incluirán las versiones utilizadas de cada una de las herramientas utilizadas en el desarrollo (ver apéndice A):

- Unity: 2019.3.0f6
- Oculus Toolkit: 20.0 [6]
- Android SDK:4.4 'KitKat'
- Blender: 2.83.0
- Photoshop: 21.0.2
- NodeJS: 12.18.0
- Express: 4.16.1
- MongoDB:

7.5.2 Hardware

Para desarrollar en Oculus Quest necesitaremos unas gafas de realidad virtual Oculus Quest. Debido a las incompatibilidades en algunas funcionalidades es importante que sean de ese modelo y marca. Por otro lado, se necesitará un móvil con Android que nos permita enlazar la cuenta de Oculus con nuestras gafas, así como el cable para conectarlas al PC.

Por último, los requisitos mínimos para utilizar Unity en nuestro ordenador serán los siguientes:

- SO: Windows 7 / 8 / 10 sólo versiones de 64 bits.
- Procesador: Core 2 Duo o superior.
- Memoria: 1 GB de RAM.
- Tarjeta gràfica: Compatible con DirectX11.

8 Implementación

8.1 Implementación del juego

Generadores de ingredientes

Los generadores de ingredientes son objetos de la escena que permiten al jugador obtener un determinado ingrediente. La principal complejidad reside en tener que instanciar un objeto dentro de otro cuando un

evento es lanzado. Para ello, se modificó el comportamiento del código encargado de coger objetos, el cual se encuentra asociado a las manos. Las clases encargadas de la interacción mano-objeto son *OVRGrabber* y *OVRGrabbable*, respectivamente. Estas clases no son propias, vienen importadas en Unity con el paquete de integración de Oculus Quest.

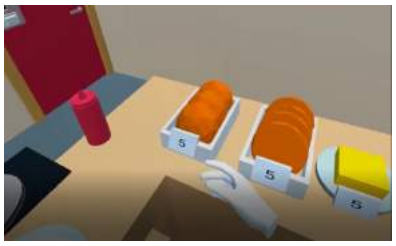

Figura 28: Generadores

Figura 29: Selección

Figura 30: Ingrediente cogido.

Al establecer una dependencia padre-hijo entre objetos, el objeto hijo queda completamente asociado al padre, siendo su posición relativa a él. En el algoritmo utilizado en dichas clases, primero se identifica qué objeto está siendo cogido por la mano. Una vez identificado, la mano pasa a ser padre de dicho objeto. Es así como se consigue el efecto de agarre. Por lo tanto, la modificación se hará al inicio, cuando se identifique el objeto. En caso que el objeto identificado sea un generador, crearemos el ingrediente correspondiente y los intercambiaremos. De esta forma, conseguiremos que la mano pase a ser padre del ingrediente, no del generador, dando la sensación que se ha sacado de dentro de éste (ver Algoritmo 1).

Algoritmo 1 Generador de Ingredientes

```
grabberCollision = ultimoObjetoAgarrado
objetoColisionado = ultimoObjetoColisionado

if objetoColisionado == GeneradorIngredientes then
    ingrediente = crearIngrediente()
    objetoColisionado = ingrediente
    grabberColision = ingredienteAgarrado
end
```

Montar plato

Una vez ponemos un ingrediente en el plato, pasa a formar parte de la hamburguesa. Por lo tanto, cuando queramos moverla por la escena, lo ideal sería que todos los ingredientes se mantuviesen juntos. Como se ha dicho en el punto anterior, una buena forma de mantener la posición de un objeto relativa a otro es establecer una relación padre-hijo. Para conseguir que la hamburguesa este unida en todo momento se ha utilizado el siguiente algoritmo (ver Algoritmo 2).

Algoritmo 2 Montar la hamburguesa

```
objetoColisionado = ultimoObjetoColisionado

if objetoColisionado == Ingrediente then
    Destruir(objetoColisionado.Rigibody)
    Destruir(objetoColisionado.OVRGrabbable)
    posicionColision = objetoColisionado.posicion
    objetoColisionado.posicion = (plato.x,posicionColision.y,plato.z)
    objetoColisionado.setParent(plato)
end
```

Además, tanto el plato como cada uno de los ingredientes tendrán una pequeña zona marcada justo encima de ellos que indicará donde va el siguiente ingrediente. La lógica de esta zona, que pasará a ser llamada como 'snap zone', también se encuentra en el mismo método (ver Algoritmo 3).

Algoritmo 3 Snap zone

```
objetoColisionado = ultimoObjetoColisionado

if plato.ingredientes == 0 then
    Desactivar(plato.snapzone)
else
    Desactivar(plato.ultimoIngrediente.snapzone)
end
Activar(objetoColisionado.snapzone)
hamburguesaActual.Añadir(objetoColisionado)
```

Figura 31: Ingredientes antes de ser colocado.

Figura 32: Snap zone

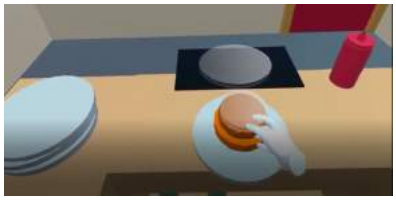

Figura 33: Nuevo ingrediente colocado.

Undo

Para realizar la funcionalidad de Undo, se decidió tener una única instancia de una clase que mantuviese una lista con cada uno de los ingredientes que se habían colocado. De esta forma, cada vez que nosotros añadimos un ingrediente a un plato, como se ha visto en el punto anterior, lo añadiremos a dicha lista. Una vez se quiere dar marcha atrás, se procederá a la ejecución del siguiente algoritmo (ver Algoritmo 4).

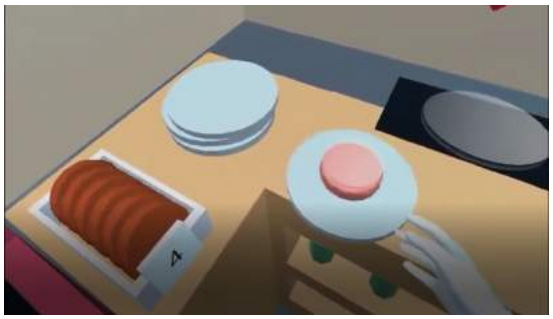

Figura 34: Antes del undo.

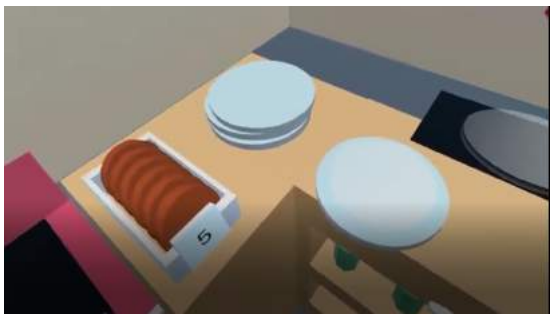

Figura 35: Despues del undo

Algoritmo 4 Undo

```
generadorIngrediente.cardinalidad += 1
Activar(hamburguesaActual.ingredientes[hamburguesaActual.ingredientes.length -2].snapzone)
Destruir(plato.ultimoIngrediente)
hamburguesaActual.Remove([hamburguesaActual.ingredientes.length -1])
```

De esta forma, mantenemos todas las variables involucradas actualizadas, dándo esa sensación de rectificación y de marcha atrás en el desarrollo del nivel.

Indicadores de selección

Una funcionalidad interesante es la de indicar al usuario en todo momento qué objeto está tocando, permitiéndole saber siempre qué objeto agarraría en caso de pulsar el botón de agarre (highlighting) (ver Figura 34 y Figura 35). Cada vez que el usuario toque un objeto con el que puede interaccionar, éste cambiará de color para hacérselo saber. Los ingredientes cambiaran al color azul, mientras que los generadores de ingredientes a negro (ver Algoritmo 5).

Algoritmo 5 Selección de ingredientes

```
objetoSeleccionado = ultimoObjetoSeleccionado
objetoColisionado = ultimoObjetoColisionado

if objetoColisionado == mano then
    if objetoSeleccionado != null then
        objetoSeleccionado.color = colorNatural(objetoSeleccionado);
    end
    objetoColisionado.color = colorSeleccion
    objetoSeleccionado = objetoColisionado
    if objetoSeleccionado == plato then
        objetoSeleccionado.childs.color = colorSeleccion
    end
end
```

De está forma, podemos tener siempre el objeto que el usuario está tocando en ese momento, dando una sensación de mayor seguridad al usuario a la hora de coger ingredientes de la escena. Además, queremos poder utilizar los mismos indicadores para los generadores. Pese a ser objetos normales, tienen ciertos comportamientos diferentes que cambian el algoritmo utilizado (ver Algoritmo 6).

Algoritmo 6 Selección de generadores

```
objetoSeleccionado = ultimoObjetoSeleccionado
objetoColisionado = ultimoObjetoColisionado

if objetoSeleccionado!= null then
    objetoSeleccionado.color = colorNatural(objetoSeleccionado)
    objetoSeleccionado.childs.color = colorNatural(objetoSeleccionado.childs)
end
objetoColisionado.color = colorSeleccion
objetoColisionado.childs.color = colorSeleccion
objetoSeleccionado = objetoColisionado
```

De igual forma, cuando retiremos el contacto de un objeto, este tendrá que volver a su color natural (ver Algoritmo 7).

Algoritmo 7 Quitar la selección

```
objetoColisionado = ultimoObjetoColisionado
objetoSeleccionado = ultimoObjetoSeleccionado

if objetoColisionado == mano then
    objetoSeleccionado.color = colorNatural(objetoSeleccionado)
    objetoSeleccionado.childs.color = colorNatural(objetoSeleccionado.childs)
    objetoSeleccionado = null
end
```

Carga de niveles del juego

La información de los niveles se encuentra almacenada de forma local. Gracias a ciertas librerías de Unity, se puede extrar la información de un JSON y almacenarla en las correspondientes clases. Para ello, se ha de tener en cuenta que los nombres han de ser exactamente iguales y que se ha de respetar la jerarquía en todo momento. A la hora de generar un pedido, simplemente escogeremos dentro de la lista de niveles cargado del JSON de forma completamente aleatoria. Cuando un nuevo tipo de pedido sea desbloqueado, se leerán los niveles correspondientes en el JSON y se añadirán a la lista de posibilidades.

La información de estos documentos queda volcada en determinadas clases del proyecto (ver sección 7.1.5) y su contenido se estructura de la siguiente forma:

- Nivel: Representa el nivel al que corresponde el pedido: 1(secuenciales),2(condicionales) o 3(iterativos).

- Tipo: Qué clase de pedido es : secuencial, condicional o iterativo.
- Img: Corresponde al directoria en el que se encuentra la imagen del pedido que se mostrará en la pantalla.
- Ingredientes: Lista de ingredientes que corresponden al pedido.
- Offset: Campo adicional utilizado únicamente en los pedidos condicionales para conocer la condición representada.

9 Conclusiones y trabajo futuro

La complejidad de realizar un juego en una plataforma como la realidad virtual es bastante elevada, tanto por todo lo que se ha de tener en cuenta como por los pocos recursos disponibles que se encuentran. Considero que se han conseguido la gran mayoría de objetivos:

- Se ha diseñado un juego que implementa una metáfora novedosa, muy pocas experiencias ofrecen una combinación así. Esa singularidad también ha hecho que fuese especialmente difícil en algunos puntos buscar ideas o fuentes de inspiración.
- Me he familiarizado mucho con la plataforma Oculus Quest, la cual nunca había probado. Además, he aprendido C# y ha utilizar Unity, sobre los cuales tampoco tenía conocimiento previo.
- Respecto al diseño, he podido modelar de forma satisfactoria muchos de los modelos 3D que tenía en mente. He aprendido a modelar y he podido presentar una escena atractiva.
- Se ha conseguido implementar el backend de forma satisfactoria, así como integrarlo al proyecto.
- Por último, se ha podido desarrollar la gran mayoría de funcionalidades, sobretodo las interacciones, en las cuales se ha puesto un especial empeño al inicio del proyecto.

Respecto al trabajo futuro, es evidente que el juego no está terminado. Quedan como puntos incompletos o a mejorar los siguientes apartados:

- Añadir más niveles, pudiendo ofrecer así una experiencia completa.
- Hacer un feedback más personalizado y concreto.
- Hacer un tutorial más interactivo y que profundice más en los conceptos de programación.
- Crear modelos que permitan decorar el entorno, dándole un aspecto más reconocible.
- Hacer pruebas con usuarios, imposibles de realizar por tiempo y por la situación sanitaria en la que se ha realizado este trabajo.

10 Bibliografía

[1] Blender. https://www.blender.org/, 2020. [Online; accessed 11-sep-2020].

[2] Express. https://expressjs.com/es/, 2020. [Online; accessed 11-sep-2020].

[3] MongoDB. https://www.mongodb.com/es, 2020. [Online; accessed 11-sep-2020].

[4] Narrator's voice. https://play.google.com/store/apps/details?id=br.com.escolhatecnologia.vozdonarrador&hl=en_US, 2020. [Online; accessed 11-sep-2020].

[5] Node.js. https://nodejs.org/es/, 2020. [Online; accessed 11-sep-2020].

[6] Oculus Integration. https://support.oculus.com/, 2020. [Online; accessed 11-sep-2020].

[7] Oculus Quest. https://www.oculus.com/quest/, 2020. [Online; accessed 11-sep-2020].

[8] Photoshop. https://www.adobe.com/es/products/photoshop.html, 2020. [Online; accessed 11-sep-2020].

[9] Unity. https://unity.com/es, 2020. [Online; accessed 11-sep-2020].

[10] Lahoti, S. Harrison Ferrone explains why C is the preferred programming language for building games in Unity. https://hub.packtpub.com/harrison-ferrone-why-c-preferred-programming-language-building-games-unity/, 2020. [Online; accessed 11-sep-2020].

[11] LuKremBo. (no copyright music) chill type beat "kitchen". https://www.youtube.com/watch?v=tUvXk4whDRA, 2020. [Online; accessed 11-sep-2020].

A Apéndice I - Guía de instalación

- Configuración Android:
  Una vez instalemos Unity, al descargar cualquiera de las versiones, tendremos que asegurarnos de seleccionar el soporte de Android:

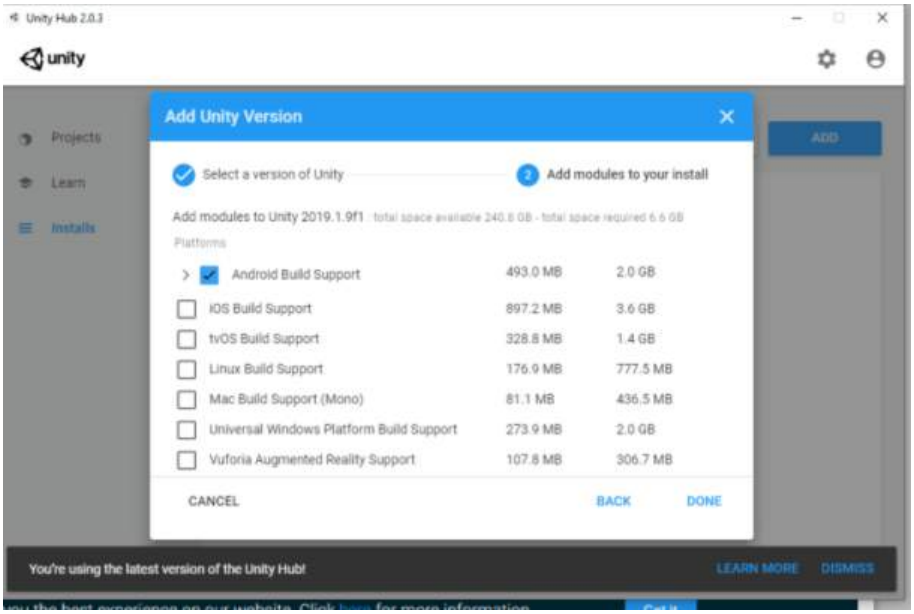

Figura 36: Selección del soporte de Android (Fuente: propia)

- Configuración:
  En este apartado se pasarán a describir los pasos más avanzados para una correcta configuración.

  - Build Settings:
    Primero, se ha de asegurar tener activada la posibilidad de ejecutar aplicaciones de origen desconocido en Oculus. Si no, simplemente no podremos probar el resultado de nuestra aplicación.

    Una vez creado el proyecto 3D, seleccionamos "Build Settings" en el menú "File". En nuestro caso, utilizaremos Android por lo que tenemos que asegurarnos de haber hecho el paso correspondiente a las SDKs.

    En la ventana que aparece, se seleccionará Switch Platform. Todas las posibles configuraciones que aparecen pueden ser predeterminadas, por lo tanto, se recomiendo dejarlas así.

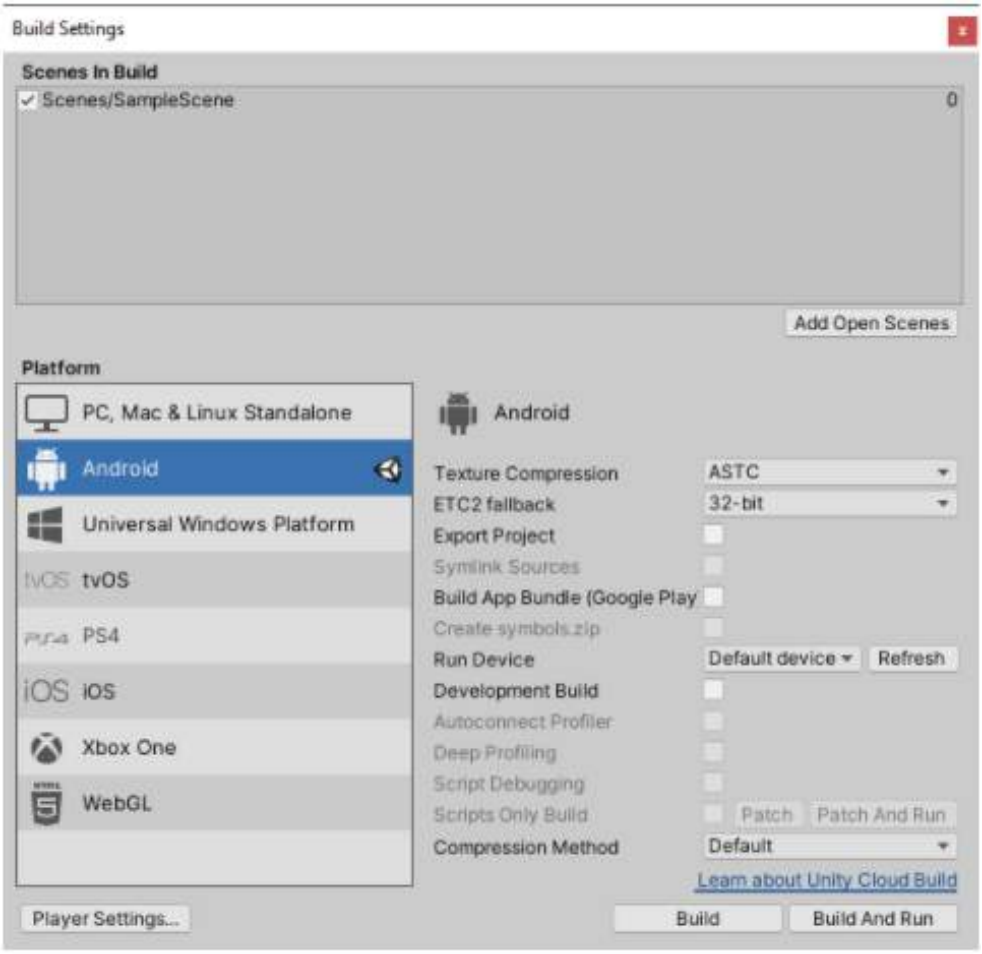

Figura 37: Selección del soporte de Android (Fuente: propia)

– Player Settings:
Proporcionar un nombre de compañía y un nombre de producto, pueden ser los que se quiera.

Eliminar Vulkan de la lista “Graphics APIs” (categoría Other Settings).

Asegurar que el nombre del paquete es correcto (corresponde con el nombre de la compañía y la aplicación).

Seleccionar la versión mínima de la API a Android 4.4 ‘KitKat’ (categoría Other Settings).

Comprobar que la integración VR está activada.

Añadir la SDK Oculus Virtual Reallity (categoría XR Settings).

Figura 38: Player Settings II(Fuente: propia)

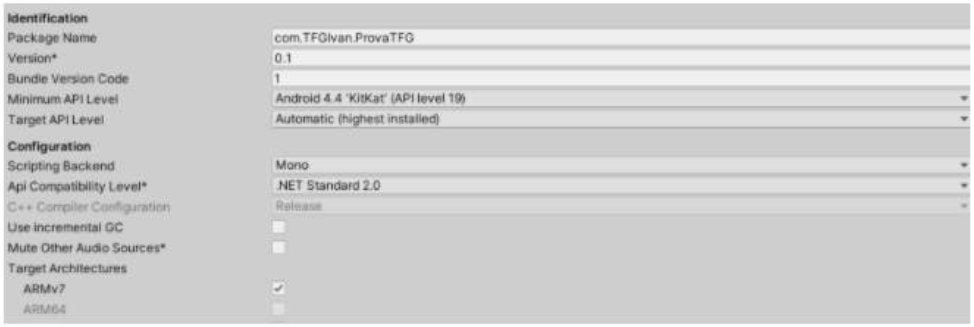

Figura 39: Player Settings II(Fuente: propia)

Figura 40: Player Settings III(Fuente: propia)

– Descarga kit integración Oculus:
  Ir a la Asset Store y buscar Oculus Integration.
  Descargar e importar el paquete al proyecto.

Una vez finalice todo el proceso, aparecerá una lista con todos los archivos que se incluirán al proyecto. Se ha de asegurar que todo está seleccionado y procedemos a importar.

- Otros paquetes:
  En Window — Package Manager seleccionar la pestaña All para ver todo el listado.Interesa especialmente añadir: Open VR y XR Legacy Input Handlers.
- Configuración Kit integración Oculus: La integración de Oculus para Unity proporciona un conjunto de soluciones muy interesantes para el desarrollador. Pese a ello, las actualizaciones del asset no necesariamente son compatibles con versiones más, o menos, recientes de Unity. Una vez el kit de integración se instale, tendremos que reiniciar el entorno. Cuidado: En ocasiones al reiniciar el IDE aparece un mensaje de actualización de paquetes o similares, el cual podemos aceptar sin problema. No obstante, cuando nos pregunte si actualizar "New Spetializer pluguin" tendremos que seleccionar la opción de eliminarlo.

  Al iniciar de nuevo, aparecerá en la barra superior un nuevo menú llamado Oculus:

  Seleccionar Tools — Remove AndroidManifest.xml.

  Seleccionar Tools — Create store-compatible AndroidManifest.xml.

  Ahora, buscamos el nuevo AndroidManifest.xml y lo abrimos en el editor. Necesitaremos comprobar que las siguientes variables tienen valores correctos:

  En lugar de "android.intent.category.INFO" sería "android.intent.category.LAUNCHER". De esta forma, cuando hagamos "Build and Run" ejecutará la APK generada en las Oculus Quest.

  Además, se ha de comprobar si la siguiente línea se encuentra en el documento:

  uses-feature android:name="android.hardware.vr.headtracking"

  android:version="1" android:required="true"

  Si todos estos cambios ya estaban presentes tampoco habrá ningún problema.

  Además, necesitaremos proporcionar una ID a nuestra aplicación. Se puede realizar en: Oculus — Avatars — Edit Settings. Con cualquier nombre, teóricamente, valdría incluso siendo aleatorio.

- Estructura de directorios del proyecto:

  - Assets: Es la carpeta donde estarán todas los scripts del proyecto. En ella, existen un conjunto de carpetas que agrupan los ficheros en función de uso. Por ejemplo, encontramos la carpeta User que guarda todas las scripts relacionadas con el usuario.
  - Resources: Cualquier recurso que se quiera utilizar en la escena (modelos 3D, sprites ...) tendrán que estar en esta carpeta. Para poder organizarlo mejor, existe una carpeta que guarda los sprites (imagenes) que se han utilizado.

– Oculus/VR/FactoryIngredients: El directorio Oculus/VR es el directorio dónde se encuentran todas las scripts del paquete de integración de Oculus Quest. En dicha carpeta, encontramos FactoryIngredients, carpeta que contiene todas las scripts relacionadas con la creación de los ingredientes (ver sección 7.1.6).

B Apéndice II - Entrevista Jesus Arbues

1- ¿Qué papel cree que tiene la tecnología, más concretamente la realidad virtual, en el futuro del sistema educativo?

La tecnología es muy importante, de hecho, está cambiando muchas formas de hacer las cosas. Hace poco se decía: si un cirujano entra en un quirófano ahora y hace 100 años, y un maestro entra en un aula ahora y hace 100 años; el maestro seguiría haciendo lo mismo y el cirujano vería muchísimo cambio. Afortunadamente, ahora estamos cambiando eso. Las tecnologías están cambiando, aunque cuesta, el sistema educativo. El claustro de profesores ya no es el que te toca en tu instituto, sino que lo tienes en todas partes, puedes tener compañeros en otros continentes. Además, los contenidos se pueden hacer de una manera mucho más agradable, rápida e intuitiva. Ahora puedes compartirlo todo.

Antes, los profesores guardaban sus apuntes como lo más preciado del mundo, no los compartían con nadie, eran suyos. Ahora, cuando se comparte todo, encuentras cosas que son muy buenas, muchísimo mejores de lo que puedes hacer tú.

La tecnología ha venido a hacer una revolución muy grande en el mundo educativo. Otra cosa es lo que queramos hacer con la educación, que es un tema mucho más delicado. La tecnología ha llegado y le ha dado la vuelta a todo, nos ha puesto en crisis.

La realidad virtual no es más que una herramienta, y ahí lo vamos a dejar. Es una herramienta que sirve para unas cosas concretas, pero no le tenemos que dar tanta importancia, no lo cambiará todo. Ni hablar, ni hablar. Puede ser tan importante, o tan buena, como un sonido 360. Es decir, depende para que la quieras puede ser muy buena, si la sabes utilizar bien.

2- ¿Por qué realidad virtual? ¿Qué características o funciones la hace algo tan interesante en la educación?

La realidad virtual va, desde los grandes aparatos como Oculus, hasta las gafas de cartón. Tenemos que saber un poco dónde nos encontramos. En el mundo de la escuela normal, pública, nunca llegaran aparatos tan buenos a todos los centros. No tenemos que pretender que todos los niños vayan con las gafas en clase durante un rato largo, sino que son cosas muy puntuales. Con las gafas, se puede ver un video de 10 minutos, una creación de alguien o algo de incluso menos duración. Por ejemplo, permitirles estar en un volcán, viendo cómo es de verdad. Al final, son cosas que no podrías hacer de forma normal. Es así de sencillo, no creo que tengamos que utilizar la realidad virtual más de lo necesario.

En las escuelas, el problema que hay, siendo yo profesor de plástica, es el hecho de crear, no sólo consumir: ¿Cómo podemos crear cosas?, ¿Qué tipo de realidad virtual podemos crear? Y ahí vienen los problemas de qué se puede hacer. Los programas de realidad virtual, como Unity o Unreal, son potentes pero muy complicados para los estudiantes. Lo único que hay bueno ahora, la pena es que es de pago, es Cospaces. Cómo mucho, sin ello, podemos hacer fotos esféricas, rutas y cosas así.

3- Me gustaría preguntarle por una frase que leí recientemente en uno de sus artículos: "En

el futuro, no conocer el lenguaje de los ordenadores será como ser analfabetos".

Es casi lo mismo que enseñar latín, no nos engañemos, es hacer que un pensamiento sea más lógico, más ordenado. Depende mucho de la situación en la que estés. Por ejemplo, existe el Scratch Junior, supongo que habrás oído hablar de Scratch, para que los críos empiecen a hacer cosas muy sencillas, bonitas y ordenadas. Luego tendríamos el normal y luego Cospaces y similares, que son programaciones más intuitivas.

Programación por programación... está bien que la gente estudie código, y sepa algo, pero eso es un resultado de lo que la sociedad está pidiendo. Van a necesitar a mucha gente con programación, para muchas cosas, pero realmente no todo el mundo tiene que estudiarla, no tiene que ser una cosa necesaria. Si que puede haber algo de pensamiento computacional, o aprender unas cadenas de: pasa esto, sino pasa esto... Está bien que se estudie, pero no es siempre necesario. Sería más importante que la gente fuese creativa, pero claro, es mucho más difícil.

4- ¿Nota que los alumnos se implican más con estas nuevas tecnologías? ¿El hecho de verse más implicados en la creación influye en el aprendizaje?

Los alumnos aprenden cuando hacen, eso es básico. Ni cuando estudian, ni cuando escuchan, ni cuando leen... los alumnos aprenden cuando hacen. Cuando explican ellos las cosas, cuando comentan lo que han hecho, ahí es cuando el aprendizaje es participativo.

Las nuevas tecnologías ayudan porque son nuevas. La realidad virtual tiene un componente de sorpresa grande: tú te pones unas gafas y de repente estás en Roma, eso es fantástico. La emoción de la sorpresa, sabiendo que también se aprende con emociones, es importante.

5- ¿Se ha encontrado con alumnos con dificultades en el uso de la realidad virtual? ¿Cómo evita esta diferencia de ritmos?

Mi experiencia, que no es la de todo el mundo, es la de utilizar esto como una ayuda puntual, en algunos casos y para cosas concretas. Lo mismo que la realidad augmentada, que puede ser un ordenador puesto a un lado del aula, con unos marcadores o lo que sea. En un momento determinado, si un niño tiene dificultades en ver una figura, lo cual suele pasar en dibujo técnico, pues va allí, coge el marcador, ve la figura en tres dimensiones y lo entiende mejor.

Los críos están abiertos a estas cosas, aunque depende también del profe y de lo que se quiera conseguir. Puedes querer que los niños salgan adelante, se hagan personas ... o puedes querer simplemente explicar el contenido, el temario, y nada más. Ahí tienes que saber dónde estás. Con estas cosas, a los críos les suele gustar que la clase no sea 50 minutos de rollo, sino una cosa puntual. Eso da más trabajo al profe. Si ves que hay más movimiento en la clase, que no están todos sentados, es un caos que, si sabes ordenarlo, los críos salen ganando y tú trabajas mucho más a gusto.

6- ¿Cómo mide usted que una determinada aplicación o taller ha sido un éxito? ¿Qué métricas utiliza?

Esto es muy difícil de saber. En educación, cuando algo es un éxito es muy difícil de saber. Puedes verlo por la cara de la gente y por la interacción que hay, por el trato humano que se establece. Es muy difícil saber esas cosas, no hay un resultado, una nota que te pongan por el taller o algo.

Yo intento hacerlo lo mejor que puedo, lo doy todo y ahí me quedo, luego ya no sé. Yo suelo dar mis apuntes, mis presentaciones, todo lo que puedo para que la gente esté más feliz. Pero no sé, no soy capaz de valorarlo.

7- ¿Por qué cree que CoSpaces es una herramienta útil y relevante para el aprendizaje?

Con esta herramienta puedes crear, no simplemente consumir, con ésta puedes crear. Y lo puedes hacer de forma bastante intuitiva, con unos resultados muy inmediatos. En el momento en que creas una cosa desde el ordenador, lo puedes tener en el móvil al momento, y lo puedes ver igual en realidad virtual. Es muy sencilla, lo más básico, pero luego tiene una parte de programación que puede ser muy potente. Volvemos con esto del pensamiento computacional, si a un crío le pone ahí para preparar alguna cosa, y él ve las posibilidades de eso, ese crío se te puede disparar. Tiene la pega de ser de pago, y ahí está el gran problema del programa. Está muy bien y es muy bueno, pero claro, la escuela pública no tiene los recursos para pagar estos recursos, al menos en este país. Yo creo que es un programa que realmente, para crear realidad virtual, va muy bien.

8- CoSpaces incorpora una forma de programar llama Blockly, ¿qué diferencias aprecia entre dicho y sistema y otros como Scratch?

Es prácticamente igual, lo que pasa es que aquí entra la tercera dimensión, no es en el plano, y puedes desplazarte teniendo en cuenta cuestiones físicas. Es bastante parecido, no están tan lejos. Se está uniformando todo esto de la programación en el mundo educativo de forma intuitiva, esto de los bloques y así, todo es muy parecido.

9- ¿Cuándo suele introducir usted el uso de la realidad virtual en la clase?

Me gusta explicar al principio, para situar las cosas, lo hago muy ordenado para que los críos no se me despisten. Al inicio de las clases, siempre hay esos 2 o 3 minutos que son de concentración máxima. En esos minutos, tienes que dar las indicaciones de todo lo que va a pasar después, en los 50 restantes, así ya saben lo que van a hacer. Son super importantes, no los puedes gastar viendo algo, por ejemplo, porque entonces luego ya no están. Siempre es mejor empezar explicando que vas a hacer, y luego, en función de una cosa u otra, montar la clase.